Semantic-based adaptive mission planning for unmanned underwater vehicles

Pedro Patrón

Doctor of Philosophy

Ocean Systems Laboratory
School of Engineering and Physical Sciences
Heriot-Watt University

April 2010

Abstract

Current underwater robotic platforms rely upon waypoint-based scripted missions which are described by the operator a-priori. This renders systems incapable of reacting to the unexpected. In this thesis, we claim that the ability to autonomously adapt the decision making process is the key to facilitating the change over from human intervention to intelligent autonomy. We identify goal-based declarative mission planning as an attractive solution to autonomous adaptability because it combines autonomous decision making with higher levels of human interaction.

Goal-based mission planning requires the use of abstract knowledge representation and situation awareness to link the prior knowledge provided by the operator with the information coming from the processed sensor data. To achieve this, we propose a semantic-based knowledge representation framework that allows this integration of prior and processed information among all different agents available in the platform. In order to evaluate adaptive mission planning techniques, we also introduce a novel metric which measures the proximity between plans. We demonstrate that this metric is better informed than previous metrics for measuring the adaptation process.

In this thesis we implement three different approaches to goal-based mission planning in order to investigate which approach is most appropriate under different circumstances. The first approach, continuous mission planning, focusses on long-term deployment. This approach is based on a continuous re-assessment of the status of the mission environment. Using our proximity metric, we evaluated this approach and show that there is a high degree of similarity between our approach and the humanly driven adaptation, both in a known static environment and in a partially-known dynamic discoverable environment. The second, service-oriented mission planning, makes use of the semantic framework to provide autonomous mission planning for the dynamic discovery of the services published by the different agents in the system. This allows platform independence, easing the manual creation of mission plans, and robustness to changes. We show that this approach produces the same plans as the baseline which was explicitly provided with the platform configuration. The last approach, mission plan repair, handles the scenario where small changes occur in the mission environment and there are limited resources for planning. We develop and deploy a mission plan repair approach within a semantic-based autonomous planning system in a real underwater vehicle. Experiments demonstrate that the integrated system is capable of providing mission adaptation for maintaining the operability of the host platform in the face of unexpected events.

Keywords: *unmanned underwater vehicles, mission flexibility, operability, decision making, mission planning, adaptive planning, autonomous planning, continuous planning, service-oriented mission planning, Markov decision process, state-space search, backward chain, plan-space search, mission plan repair, plan proximity, situation awareness, machine awareness, semantic knowledge, service discovery, dynamic adaptation, robust execution, service-oriented architecture*

To Dad
for reaching the top from the roots

Every person ought to have a child, to plant a tree and to write a book before they die.

– José Martí

Acknowledgements

It has been six years since I came to Edinburgh to join the 'Scottish Clan' of the Ocean Systems Laboratory and SeeByte Ltd. Both teams have grown so big since my arrival that they are now too many people to mention them all individually. They have showed me how to enjoy wet Scottish days and having cold hands during the many days of trials that we shared together working on the challenging discipline of underwater robotics. This thesis only covers a small part of the work that I have done in these years spent in Edinburgh. The experience gained and the relationships I have made during this time have been invaluable. Thank you all.

Thank you also to my supervisor Prof. David Lane for always finding resources for my work and for supporting me even during his shortages of bandwidth. Prof. Yvan Petillot deserves a special thank you for knowing how to combine being a great friend while still maintaining a fruitful professional relationship.

Finally, I would like to thank my supporters backstage. To my family and friends here and at home, for always being there. Thank you to Lexi for following me to Edinburgh, for our fruitful discussions in the late evenings, for providing such amazing regexp to my humble scripts, and for always knowing how to keep a phase-shift between the highs and lows of my PhD and hers. But above all I want to thank her for giving me the best degree of my life: our baby 'croqueta' Alba.

It is time to plant a tree now.

ACADEMIC REGISTRY
Research Thesis Submission

Name*:*	Pedro Patrón		
School/PGI:	School of Engineering and Physical Sciences		
Version: *(i.e. First, Resubmission, Final)*	Final	Degree Sought (Award **and** Subject area)	PhD

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1) the thesis embodies the results of my own work and has been composed by myself
2) where appropriate, I have made acknowledgement of the work of others and have made reference to work carried out in collaboration with other persons
3) the thesis is the correct version of the thesis for submission and is the same version as any electronic versions submitted*.
4) my thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
5) I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate*:*		Date:	

Submission

Submitted By *(name in capitals):*	
Signature of Individual Submitting:	
Date Submitted:	

For Completion in Academic Registry

Received in the Academic Registry by *(name in capitals):*			
Method of Submission *(Handed in to Academic Registry; posted through internal/external mail):*			
E-thesis Submitted (***mandatory for final theses from January 2009)***			
Signature:		Date:	

Table of Contents

List of Figures

List of Tables

Glossary

***Quotation**, n: The act of repeating erroneously the words of another.*

– Ambrose Bierce

- **ABox** is an 'assertion component' – a fact associated with a terminological vocabulary within a knowledge base or ontology. See also *TBox* and *Ontology*.
- **Acceptance** is the act or process of favourable reception or approval.
- **Adaptability** is the property of a system to change in order to adjust itself to the new conditions.
- **Admissible** a heuristic is said to be admissible if it never overestimates the cost of reaching the goal.
- **Affordability** is the property of the system that relates with having the financial means for the system or bear its cost.
- **Agent** is any external module, algorithm or process with access to the knowledge base framework.
- **Application Ontology** is an ontology containing relevant information for a particular application or agent. This ontology is domain independent and context dependent. See also *Core Ontology*.
- **Architecture** is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships between them.
- **Attritable** is used to define an unmanned system that is somewhat survivable. Loss of the unmanned system will have moderate cost and/or operational impact,

but the operational benefits outweigh the potential risks. See also *expendable* and *survivable*.

- **Automatic** means that a system will do exactly as programmed, it has no choice. See also *Autonomous*.
- **Autonomous** means that a system has a choice to make free of outside influence. See also *Automatic*.
- **Availability** is considered as the amount of time the system is available to the user at the time it is needed.
- **Capability** is the ability to perform actions.
- **Context** describes the surroundings, circumstances, environment, background or settings which determine, specify, or clarify the meaning of an agent.
- **Component** provides a unique functional capability for the unmanned system. A component resides wholly within a node.
- **Core Ontology** is an ontology containing common information to all service-oriented agents in the platform. This ontology is domain dependent and context independent. See also *Application Ontology*.
- **Data** is perceived knowledge about a specific subject or situation: fact (used in plural). See also *Information* and *Knowledge*.
- **Domain** is the territory over which rule or control is exercised. For this project, the core ontology will be defined for the unmanned systems domain.
- **Expendable** is used to define an unmanned system that is minimally survivable. Loss of the unmanned system has minimal cost and operational impact, the unmanned system can be quickly replaced or is not critical to operational success. See also *attritable* and *survivable*.
- **Hard Real Time** is defined when the processing time (latency) of the process must be guaranteed. This latency may be defined in terms of either minimum and maximum times, or only a maximum. See also *Real Time*.
- **Information** is knowledge composed by the sum of what has been perceived, discovered, or inferred. See also *Data* and *Knowledge*.

- **Interoperability** is the ability of two or more subsystems to exchange information and to use the information that has been exchanged.
- **Knowledge** is which is known. See also *Information* and *Data*.
- **Knowledge Base** is an ontology. See *Ontology*.
- **Knowledge Base Repository** is the agent dealing with the storage of the ontology ABox. When it only stores data, it can also be known as World Model.
- **Knowledge Store** is a knowledge base repository. See *Knowledge base repository*.
- **Metaknowledge** is knowledge about a preselected knowledge. It is a cluster of definitions and methods aiming to guide the gathering of the pertinent knowledge with regard to an activity. See also *Metadata* and *Ontology*.
- **Metadata** is data about data. See also *Metaknowledge* and *Ontology*.
- **Mission Planing** is the process by which a human operator or AI algorithm devises tactical goals, a route (general or specific), and timing for one or more unmanned vehicles. Considerations include terrain, threat, weather, and location of friendly forces, fire support, and mission modules. The mission planning process may be accomplished on a computer or operator control unit for downloading to the unmanned vehicle.
- **Ontology** (aka: Knowledge base) is a representation of a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to define the domain. It is a special kind of database for knowledge management. It provides the means for the computerized collection, organization, and retrieval of knowledge. Together Abox and Tbox statements make up a knowledge base.
- **Operability** of a system is defined as being such that use or operation is possible.
- **Planning** is the reasoning side of acting. It is an abstract explicit deliberation process that chooses and organizes actions by anticipating their expected outcomes.

- **Real time** (aka: on-line) is a property of a process that allows it to react to data and messages from sensors and operators as they occur, and can be contrasted with off-line, which refers to processes that respond to stored data.
- **Recoverability** is the measure of time taken to recover from damage and/or the extent to which it is possible to restore fully or partially the capability of the unmanned system.
- **Reliability** is the probability that an item will perform its intended function for a specified time under stated conditions.
- **Remotely operated vehicles (ROVs)** is the term used for vehicles that are remotely controlled by humans.
- **Requirement** is a need, a demand or a constraint.
- **Robustness** a system is said to be robust if it has demonstrated an ability to recover gracefully from the whole range of exceptional inputs and situations in a given environment.
- **Service-Oriented Agent** is an agent providing a service or capability to the architecture.
- **Service-Oriented Architecture** defines a system made of up distributed capabilities that are self-contained, loosely coupled, and have well defined interfaces.
- **Soft real time** is defined when latency is not guaranteed. The only requirement is that processing rates are no less than data input rates when averaged over a sufficiently long time-period. See also *Real Time*.
- **Survivability** is the capability of a platform to avoid or withstand a man-made hostile environment.
- **Survivable** is used to define an unmanned system that is highly survivable. Loss of the unmanned system will have a significant cost and/or operational impact. See also *Expendable* and *Attritable*.
- **Susceptibility** is the probability of being hit. The inability of an unmanned system to avoid the threats in a man-made hostile environment.

- **System** is a logical grouping of subsystems. It therefore provides a functional grouping for the full robotic or unmanned capability.
- **Subsystem** performs one or more unmanned system functions as a single localized entity within the framework of the system. A subsystem shall provide one or more capabilities.
- **TBox** is a 'terminological component' – a vocabulary associated with a set of facts ABox within a Knowledge base or Ontology. See also *ABox* and *Ontology*.
- **Uninhabited vehicles** are vehicles with no human crew inside. They can be remotely operated or autonomous vehicles.
- **Unmanned vehicles** is the term normally used for truly autonomous vehicles.
- **Validation** method to understand if the system is built as described.
- **Verification** method that established the correctness of a theory.
- **Vulnerability** is the probability of being damaged, once it has been hit. The inability of an aircraft to withstand a man-made hostile environment.
- **World Model** See *Knowledge base repository*.

Acronyms

Acronym	*Meaning*
AI	Artificial Intelligence
ASCII	American Standard Code for Information Interchange
ATR	Automatic Target Recognition
AUV	Autonomous Underwater Vehicle
BAUUV	Battlespace Access for Unmanned Underwater Vehicles
CAC	Computer-Aided Classification
CAD	Computer-Aided Detection
COI	Competition of Ideas
COTS	Commercial off-the-self
DoD	Department of Defense (US)
DSTL	Defence Science and Technology Laboratory
DTC	Defence Technology Centre
DTIC	Defence Technology & Innovation Centre
DVL	Doppler Velocity Log
FDDR	Failure, Detection, Diagnosis and Recovery
GPS	Global Positioning System
HWU	Heriot-Watt University
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IRM	Inspection Repair Maintenance
ISR	Intelligence Surveillance Reconnaissance
MDP	Markov Decision Process
MoD	Ministry of Defence (UK)
PDDL	Plan Domain Definition Language
OODA	Observe, Orient, Decide & Act

(continued)

Acronym	*Meaning*
OSL	Ocean Systems Laboratory
RDG	Reactive Data Gathering
RPC	Remote Procedure Call
ROV	Remotely Operated Vehicle
SA	Situation Awareness
SAS	Synthetic Aperture Sonar
SE	Synthetic Environment
SEAS	Systems Engineering for Autonomous Systems
SLAM	Simultaneous Localization and Mapping
SOA	Service-Oriented Agent
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UUV	Unmanned Underwater Vehicle

Table of Notation

Notation	*Meaning*	*First Occurrence*
	Symbols	
$\uparrow a$	increment of a	Eq. 1.1
$\downarrow a$	decrement of a	Eq. 1.2
$a \rightarrow b$	transition function, from a to b	Sec. 4.3.2
$a \Rightarrow b$	implication, if a then b	Eq. 1.1
$a \iff b$	a if and only if b	Sec. A.1
$a \wedge b$	conjunction operator, a and b	Eq. 1.1
$a \vee b$	disjunction operator, a or b	Eq. 1.3
a'	modified a	Def. 3.2.1
$\hat{a}$	normalized a	Eq. 3.4
$\bar{a}$	complement of a	Eq. A.2
$\sim a$	approximation of a	Sec. 1.2.3.3
$\tilde{a}$	estimated or predicted a	Eq. 4.8
$\dot{a}$	observed or measured a	Alg. 4.4.1
$\{a, b, \ldots\}$	set of elements	Sec. 4.3.1
$\langle a, b, \ldots \rangle$	list of elements	Sec. 4.3.1
$(a, b, \ldots)$	tuple of elements	Def. 3.2.1
	Greek letters	
α	balance factor	Eq. 3.7
β	discount factor	Eq. 4.8
γ	cost of execution	Sec. 4.3.2.3
Γ	uncertainty matrix	Eq. 4.1
δ	reward value	Sec. 4.3.2.2
Δ	plan matrix	Eq. 4.8
η	plan matrix element	Eq. 4.11

(continued)

Notation	*Meaning*	*First Occurrence*
Θ	domain space	Sec. 1.4
κ	number of changes to a plan	Eq. 3.11
λ	operator goal reward	Sec. 4.3.2.2
μ	number of ground action insertions	Eq. 3.11
π	plan, mission plan	Sec. 1.4
Π	mission environment	Sec. 1.4
ρ	Spearman's rho rank correlation coefficient	Sec. 3.4
σ	payoff function, utility function	Eq. 4.7
ς	plan matrix row	Eq. 4.11
Σ	domain model	Sec. 1.4
τ	continuous plan length	Eq. 4.8
υ	number of ground actions deletions	Eq. 3.11
ϕ	passive action	Sec. 4.3.2.5
ψ	partial plan	Fig. 6.2
φ	progression function	Sec. 5.3.2.1
ω	number of permutations	Eq. 3.11
Ω	problem model	Sec. 1.4
	Latin letters	
a	action	Sec. 4.3.1
A_V	set of actions	Sec. 4.3.1
b	service	Eq. 5.7
B	set of services	Eq. 5.7
c	class	Def. 3.5.2
C	set of classes	Sec. 4.3.1
d_a	action difference	Eq. 3.26
d_o	object difference	Eq. 3.18
d_p	resource difference	Eq. 3.23
D	plan stability	Def. 3.3.1
D_c	class distance	Eq. 3.14
D_r	fact distance	Eq. 3.22
D_g	ground action distance	Eq. 3.25
D_o	object distance	Eq. 3.17
D_p	plan difference	Def. 3.3.2

(continued)

Notation	*Meaning*	*First Occurrence*
D_s	state difference	Def. 3.3.3
D_v	attribute distance	Eq. 3.20
e	execution of a ground action	Sec. 4.3.2
e_p	number of extra ground actions in a plan	Def. 3.3
f	function	Sec. 4.3.1
g	ground action	Sec. 4.3.1
G	goal state, final state	Def. 3.2.1
G_O	set of ground actions	Sec. 4.3.1
h	heuristic	Sec. 5.3.2.1
i	binary state information	Sec. 4.3.2
I	original state, initial state	Def. 3.2.1
l	depth level of a class hierarchy tree	Eq. 3.15
m	state size, number of proposition facts in a state	Def. 3.5
m_p	number of missing actions in a plan	Def. 3.3
M	total number of proposition facts in a domain	Def. 3.5
n	plan length, number of ground actions in a plan	Def. 3.2.2
$\mathbb{N}_0$	set of natural numbers including 0	Sec. 4.3.2
o	object	Def. 3.5.3
O_C	set of objects	Sec. 4.3.1
p	proposition	Def. 3.5.5
P	planning strategy	Def. 3.3.5
P_V	set of propositions	Sec. 4.3.1
PP_α	plan proximity for a given balance factor α	Def. 3.3.4
q	step in time	Sec. 4.3.2
Q_O	set of proposition fact goals	Sec. 4.3.1
r	proposition fact	Def. 3.5.5
R_O	set of proposition facts	Sec. 4.3.1
$\mathbb{R}$	set of real numbers	Sec. 4.3.1
$\mathbb{R}_+$	set of positive real numbers	Sec. 4.3.2.3
s	state	Def. 3.2.3
S	set of states	Sec. 5.3.2.1
SD_s	semantic state difference	Eq. 3.28
SD_p	semantic plan difference	Eq. 3.30

(continued)

Notation	*Meaning*	*First Occurrence*
SPP_{α}	semantic plan proximity	Eq. 3.32
t	time	Sec. 1.4
T	planning horizon	Sec. 4.3.2
u	plan candidate	Sec. 4.3.2
U	set of ground action candidates	Alg. 4.4.3
v	attribute, variable	Def. 3.5.4
V_C	set of variables	Sec. 4.3.1
w	state candidate	Eq. 5.3
W	set of state candidates	Eq. 5.3
x	binary string representation of a state	Def. 3.5

List of Publications

This research has produced the following publications [1]:

Journals, Book Chapters and Magazines

- Miguelanez, E., Patrón, P., Brown, K., Petillot, Y. R., and Lane, D. M. (2010). Semantic knowledge-based framework to improve the situation awareness of autonomous underwater vehicles. *IEEE Transactions on Knowledge and Data Engineering (In Press)*, PP(99)

- Patrón, P. (2009). Embedded knowledge and autonomous planning. *Sea Technology Magazine, ISSN. 0093-3651*, 50(4):101

- Pêtrès, C., Pailhas, Y., Patrón, P., Evans, J., Petillot, Y., and Lane, D. (2009). *Underwater Vehicles*, chapter 21:Trajectory Planning for Autonomous Underwater Vehicles, pages 399–416. I-Tech, ISBN 978-953-7619-49-7

- Evans, J., Patrón, P., Smith, B., and Lane, D. (2008). Design and evaluation of a reactive and deliberative collision avoidance and escape architecture for autonomous robots. *Journal of Autonomous Robots*, 3:247–266

- Pêtrès, C., Pailhas, Y., Patrón, P., Petillot, Y. R., Evans, J., and Lane, D. M. (2007). Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics*, 23(2):331–341

- Patrón, P., Smith, B., Pailhas, Y., Capus, C., and Evans, J. (2007b). Strategies and sensor technologies for UUV collision, obstacle avoidance and escape. *Journal of the Undersea Defence Technology Forum*, 1:31–36

[1] shown in reverse chronological order

Conferences and Workshops

- Patrón, P. and Birch, A. (2009). Plan proximity: an enhanced metric for plan stability. In *Workshop on Verification and Validation of Planning and Scheduling Systems, 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 74–75, Thessaloniki, Greece

- Patrón, P., Lane, D. M., and Petillot, Y. R. (2009a). Continuous mission plan adaptation for autonomous vehicles: balancing effort and reward. In *4th Workshop on Planning and Plan Execution for Real-World Systems, 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 50–57, Thessaloniki, Greece

- Patrón, P., Lane, D. M., and Petillot, Y. R. (2009b). Interoperability of agent capabilities for autonomous knowledge acquisition and decision making in unmanned platforms. In *Proceedings of the IEEE International Conference Oceans Europe (Oceans Europe'09)*, Bremen, Germany

- Petillot, Y. R., Sotzing, C., Patrón, P., Lane, D. M., and Cartwright, J. (2009). Multiple system collaborative planning and sensing for autonomous platforms with shared and distributed situational awareness. In *Proceedings of the AUVSIs Unmanned Systems Europe*, La Spezia, Italy

- Patrón, P., Lane, D. M., and Petillot, Y. R. (2009c). Situation-aware mission planning using distributed service oriented agents in autonomous underwater vehicles. In *International Symposium on Unmanned Untethered Submersible Technology*, Durham, NH, USA

- Evans, J., Patrón, P., Privat, B., Johnson, N., and Capus, C. (2009). AUTOTRACKER: Autonomous inspection capabilities and lessons learned in offshore operations. In *Proceedings of the IEEE International Conference Oceans (Oceans'09)*, Biloxi, MS, USA

- Patrón, P. and Petillot, Y. R. (2008). The underwater environment: A challenge for planning. In *Proceedings of the Conference of the UK Planning Special Interest Group (PlanSIG'08)*, Edinburgh, UK

- Patrón, P., Miguelanez, E., Cartwright, J., and Petillot, Y. R. (2008a). Semantic knowledge-based representation for improving situation awareness in service

oriented agents of autonomous underwater vehicles. In *Proceedings of the IEEE International Conference Oceans (Oceans'08)*, Quebec, Canada

- Patrón, P., Miguelanez, E., Petillot, Y. R., and Lane, D. M. (2008b). Fault tolerant adaptive mission planning with semantic knowledge representation for autonomous underwater vehicles. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08)*, pages 2593–2598, Nice, France

- Patrón, P. and Lane, D. M. (2008). Adaptive mission planning: The embedded OODA loop. In *Proceedings of the Conference of Systems Engineering for Autonomous Systems from the Defence Technology Centre (SEAS-DTC'08)*, Edinburgh, UK

- Patrón, P., Miguelanez, E., Petillot, Y. R., Lane, D. M., and Salvi, J. (2008c). Adaptive mission plan diagnosis and repair for fault recovery in autonomous underwater vehicles. In *Proceedings of the IEEE International Conference Oceans (Oceans'08)*, Quebec, Canada

- Johnson, N., Patrón, P., and Lane, D. M. (2007). The importance of trust between operator and AUV: Crossing the human/computer language barrier. In *Proceedings of the IEEE International Conference Oceans Europe (Oceans Europe'07)*, Aberdeen, UK

- Cartwright, J., Patrón, P., Evans, J., and Lane, D. M. (2007). Distributed ontological world model for autonomous multi vehicle operations. In *Proceedings of the Conference of Systems Engineering for Autonomous Systems from the Defence Technology Centre (SEAS-DTC'07)*, Edinburgh, UK

- Patrón, P., Evans, J., and Lane, D. M. (2007a). Mission plan recovery for increasing vehicle autonomy. In *Proceedings of the Conference of Systems Engineering for Autonomous Systems from the Defence Technology Centre (SEAS-DTC'07)*, Edinburgh, UK

- Davis, B. C., Patrón, P., Arredondo, M., and Lane, D. M. (2007a). Augmented reality and data fusion techniques for improved testing & enhanced situational awareness of the underwater domain. In *Proceedings of the IEEE International Conference Oceans Europe (Oceans Europe'07)*, Aberdeen, UK

- Davis, B. C., Patrón, P., and Lane, D. M. (2007b). An augmented reality architecture for the creation of hardware in the loop and hybrid simulation test scenarios for unmanned underwater vehicles. In *Proceedings of the IEEE International Conference Oceans (Oceans'07)*, Vancouver, Canada

- Evans, J., Sotzing, C., Patrón, P., and Lane, D. (2006). Cooperative planning architectures for multi-vehicle autonomous operations. In *Proceedings of the Conference of Systems Engineering for Autonomous Systems from the Defence Technology Centre (SEAS-DTC'06)*, Edinburgh, UK

- Patrón, P., Evans, J., and Lane, D. M. (2006b). Fault tolerant decision making for unmanned vehicles. In *Proceedings of the Conference of Systems Engineering for Autonomous Systems from the Defence Technology Centre (SEAS-DTC'06)*, Edinburgh, UK

- Patrón, P., Evans, J., Brydon, J., and Jamieson, J. (2006a). AUTOTRACKER: Autonomous pipeline inspection: Sea trials 2005. In *Proceedings of the World Maritime Technology Conference - Advances in Technology for Underwater Vehicles (WMTC'06)*, London, UK

- Patrón, P. and Tena-Ruiz, I. (2006). A smooth simultaneous localisation and mapping solution to improve situational awareness, mission planning and replanning for AUVs. In *Proceedings of the World Maritime Technology Conference - Advances in Technology for Underwater Vehicles (WMTC'06)*, London, UK

- Patrón, P., Smith, B., Pailhas, Y., Capus, C., and Evans, J. (2005). Strategies and sensors technologies for UUV collision, obstacle avoidance and escape. In *7th Unmanned Underwater Vehicle Showcase (UUVS'05)*, Southampton, UK

- Pêtrès, C. and Patrón, P. (2005). Path planning for unmanned underwater vehicles. In *Workshop on Planning and Learning in A Priori Unknown or Dynamic Domains, 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, Edinburgh, UK

Chapter 1

Introduction

The cure for anything is salt water - sweat, tears, or the sea.

– Isak Dinesen

1.1 Introduction

Robotics is of crucial importance to marine industries. As the need for deeper and more comprehensive access to the underwater world grows, so does the need for autonomy. We address this problem in our thesis by proposing a semantic-based adaptive mission planning framework. This framework combines the advantages of autonomous decision making with high-level interactions with the operator. We show that this approach raises the adaptability and the recoverability of unmanned underwater robots. This chapter presents the research challenges, objectives and contributions of the thesis.

This chapter describes the challenges faced by robotic systems when accessing the underwater domain. Then, we focus on the specific challenge of providing autonomous decision making for adaptive mission management. Overcoming this challenge is expected to improve the operability of underwater robotic systems, and is the main motivation of this thesis. Subsequently the research objectives are extracted by looking at different scenarios related to several underwater applications. Then we analyse current approaches tackling these objectives and we propose a new decision making framework to overcome the challenge of adaptive mission management. This framework becomes the basis of this research. The chapter finishes by describing the main contributions of this thesis and by introducing the contents of the rest of the chapters.

1.2 The Underwater Environment

1.2.1 Motivation

In the last decades, the oceans have been the focus of unprecedented interest. Although they cover 71% of the Earth's surface, humankind has sent more astronauts to the Moon than scientists to the deepest parts of our seas. Governments and industry have now become more and more interested in understanding and managing our planet. They have realised how important the underwater regions, two thirds of the total Earth's surface, are. Nowadays, it is not only the need to discover, but also to observe, to map, and to protect our oceans that motivates further exploration of underwater regions.

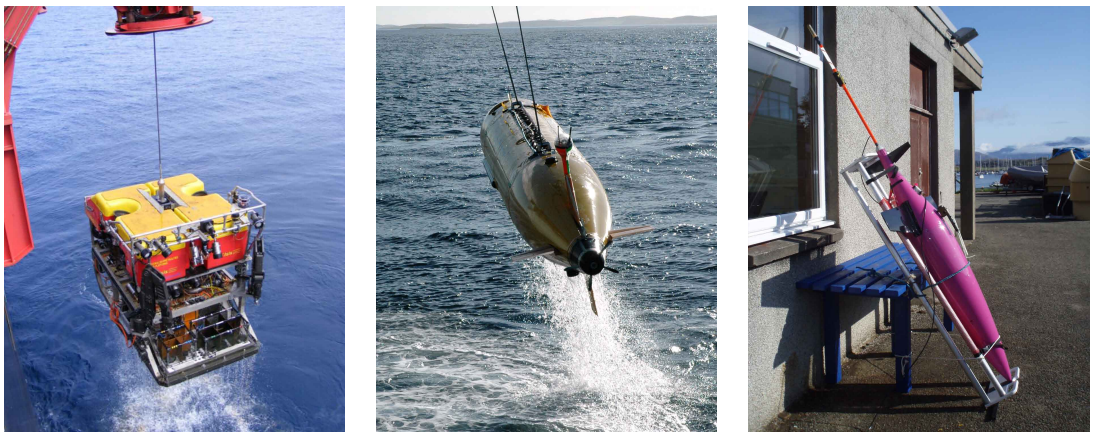
Unfortunately, access to these regions is not straightforward. The underwater domain is a hostile environment for humans and human technology. It can challenge some of the capabilities that are now taken for granted in other domains such as the Earth's surface, the atmosphere or the outer Space. Some of the most representative and specific challenges underwater are high pressure, corrosion and signal processing issues related to data communications and sensing.

Even under such challenges, several maritime disciplines still require access to this environment. The most relevant ones are:

- **Oceanography:** Scientists need to gain access to the most remote parts of the oceans, from deep trenches (Todo et al., 2005) to fresh water lakes under the polar ice caps (Bell et al., 2007). They have to collect information in order to be able to understand issues such as life under extreme conditions, climate change, the melting of the polar ice caps, and to forecast weather conditions, hurricanes and tsunamis.

- **Energy and Mining industry:** In current offshore oil fields the tasks of Inspection, Repair and Maintenance (IRM) comprise up to 90% of the offshore field activity (Billingham, 2004). This inspection is currently dictated by availability of surface vessels and weather conditions. Additionally, the deep sea is still unexploited. Gaining access to waters deeper than the continental shelf can provide access to new sources of minerals and energy (Murton, 2000).

- **Military:** Two of the main priorities of current Navy operations are maintaining clear access to ship passages (Dobeck et al., 1997) and protecting vessels, harbours and coastal waters (Reed et al., 2006a). Achieving these capabilities without compromising personnel safety is still unsolved.



a) b) c)

Figure 1.1: a) Recovery of ISIS ROV (courtesy of National Oceanographic Centre, UK). b) Geosub AUV recovery after finishing a mission (courtesy of SeeByte Ltd., UK). c) Talisker SeaGlider (courtesy of Scottish Association for Marine Science, UK).

1.2.2 Robotic Platforms

For all these disciplines, robotic platforms are proven to be very useful in de-risking human activity in the hostile underwater domain. Underwater vehicles have become a standard tool for data gathering for maritime applications (Griffiths, 2003). Unmanned underwater vehicles (UUVs) can be classified into *Remotely Operated Underwater Vehicles* (ROVs), *Autonomous Underwater Vehicles* (AUVs) and *Underwater Gliders* (see Figure 1.1). There is currently a total fleet worldwide of over 4000 unmanned underwater vehicles (Newman et al., 2007). They differ with respect to their power capability, endurance and the complexity of the tasks that they have been designed for:

- **Remotely Operated Vehicles (ROVs):** They are able to remotely connect humans to the underwater world. They enable real-time operator-driven underwater sensor data gathering and actuator intervention. A tether connection to the surface vessel provides them unlimited power and communication. For this reason, they are limited to vessel availability and require complicated management of the tether system.

- **Autonomous Underwater Vehicles (AUVs):** They are tetherless and therefore run on limited power supply, such as batteries. They are independent of surface parameters, such as vessel availability or weather, but they are currently limited to survey or inspection tasks that do not require intervention.

- **Underwater Gliders:** They are a special case of AUVs. They have got a very low power consumption payload and do not have propulsion system. In consequence, they are suitable for long-term long-distance data gathering. The lack of propulsion system means that they have limited control of their destination.

Recent hybrid combinations of these platforms (Chardard and Copros, 2002; Caffaz et al., 2009) indicate that the distinction between these three classes will become less clearly defined in future years.

1.2.3 Challenges

The main challenges that these robotic systems have to deal with underwater are power, communication, perception, navigation, and decision making. We presented a survey study identifying these challenges in Patrón and Petillot (2008).

1.2.3.1 Power

Robots are highly dependant on their battery life in a domain without possibility of extending it from other external sources (Griffiths, 2005b).

The continuous reduction of hardware power demands with the development of micro and nano electronics (Lynn, 2001), and the appearance of reliable long-range battery cells and fuel cells (Yamamoto et al., 2004) are steadily mitigating this problem.

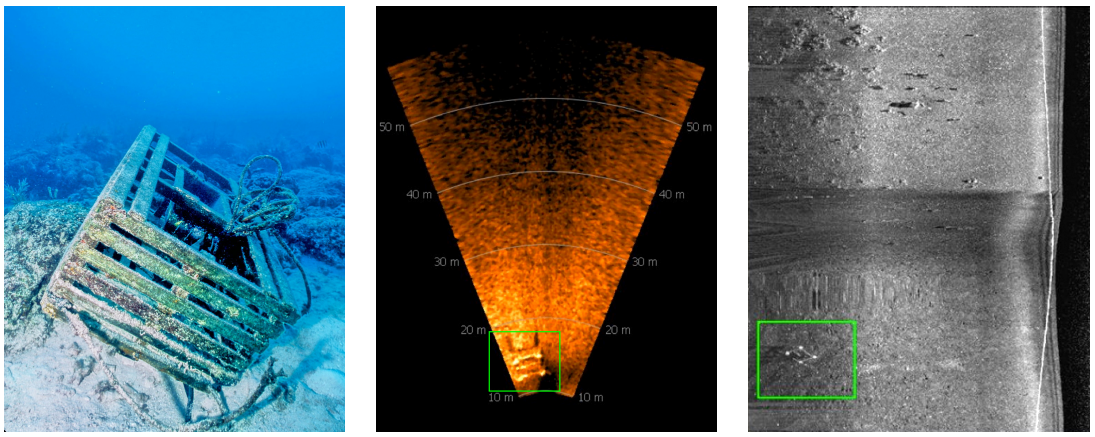
1.2.3.2 Communication

Sound is the common media used for communicating underwater. In this media, low bandwidth, long delays and high-power requirements of emitters and receivers impose many restrictions (Kilfoyle and Baggeroer, 2000; Stojanovic, 2003).

Solutions that use frequency shifting, phase shifting and compression techniques (Yu, 2000; Freitag et al., 2005) and novel radio frequency approaches (Rhodes and Hyland, 2009) have been recently developed in order to overcome these limitations.

1.2.3.3 Perception

Visual methods are poor underwater as they depend on the amount of light reaching the objects. Due to refraction and absorption, only 1% of the natural light on the surface reaches as deep as $100m$. Even if artificial light is provided, scatter noise restricts the



a) b) c)

Figure 1.2: Different perceptions of a lobster cage: a) underwater camera image at 2m range, b) acoustic image from a 450kHz forward looking sonar at 15m range and, c) acoustic image from a 900kHz sidescan sonar at 50m range (shown inside the box).

vision range (see Figure 1.2.a). This noise is produced by suspended biomass and the water molecules themselves deflecting light and reducing contrast.

On the other hand, acoustic methods are affected by temperature, pressure and salinity. These factors make them notably noisy. Also, their range is inversely related to the frequency and normally considerably reduced (see Figure 1.2.b and Figure 1.2.c).

Novel Synthetic Aperture Sonar (SAS) devices are able to provide near-to-visual constant resolution over long ranges (Bellettini and Pinto, 2002). Resolutions of $\sim 4cm$ over $200m$ ranges have been recently achieved (Hagen, 2008). These results allow the reduction of false positives during the data analysis. The technique uses the vehicle motion to create a long synthetic array. This makes these devices highly dependent on the accuracy and precision of the navigation provided.

Ultimately, raw sensor data has to be processed into concepts in order to be able to extract the information from the environment. These processes of conceptualisation come with numerous false positives. An important element of this thesis is how to leverage concepts on-line and link them with prior information for mission planning.

1.2.3.4 Navigation

Navigating in underwater environments is one of the major challenges for robotics, as it entails a high level of uncertainty. Trajectory planning solves the problem of navigating

from one point to another. Although the focus of this thesis is mission planning, which deals with the next level up of autonomy, the two are intimately linked. We have much previous work on trajectory planning, which has provided context for this thesis. Two steps are required to navigate: geolocating the platform and orienteering through the sensed obstacles.

Geolocating a vehicle on the Earth's surface can be accurately performed by using Global Positioning System (GPS) receivers. But GPS does not work underwater as the radio signals on which it depends cannot pass through water. Additionally, existing underwater maps are still fairly inaccurate. Thus, *dead reckoning* is the standard technique used underwater. It involves use of Inertial Measurement Units (IMUs), a combination of accelerometers and gyroscopes. However, abbe error – magnification of angular error over distance – adversely affects dead reckoning.

Inertial Navigation Systems (INS) integrate accelerations and rates to provide a Kalman-filtered navigation solution (Kalman, 1960). They use inputs from acoustic transmitters and receivers to pin-point the location of the platform. However, the range of these systems is limited to no more than a few nautical miles. Thus, restricting the autonomy of the vehicle.

A better approach involves aiding the INS with a Doppler Velocity Log (DVL) sensor. This sensor measures the displacement rate over the seabed. However, no matter how accurate the sensors are, the errors in these systems grow with time. As a consequence, the platform becomes progressively lost unless it is able to obtain external references from acoustic devices or from a GPS receiver on the surface. Furthermore, this uncertainty propagates to the geolocation of the observed elements in the environment during the mapping process.

The technique of Simultaneous Localisation And Mapping (SLAM) is a promising alternative (Durrant-Whyte and Bailey, 2006). Using SLAM, a platform maps the environment and uses the map to localise itself in it. The map can be georeferenced if the platform maintains an estimate of its absolute position when the process of mapping is started. It is possible to smooth the SLAM solution in order to create better maps of the environment. Hence, trajectory planning systems are able to accurately locate the sensed obstacles, as we demonstrated in Patrón and Tena-Ruiz (2006).

Once the navigation and mapping error have been bounded, a collision-free efficient trajectory is required to orient the platform. Traditional approaches to collision avoidance and escape have been purely reactive (Pang et al., 2003). Recently, approaches based on rules of collision (Benjamin et al., 2006) and our approach to obsta-

cle avoidance and escape scenarios (Patrón et al., 2007b; Evans et al., 2008) have been deployed.

At the trajectory planning level, these approaches use lifelong planning incremental search methods (Stentz, 1994; Koenig et al., 2004). These methods have been found to decrease computation time while still locating the shortest trajectory between the vehicle and its destination. Instead of starting from scratch every time a trajectory must be adapted, they reuse data found in previous searches to save on computation time. Wave propagation techniques can also provide with smoother and shorter trajectories than classic discrete search approaches. In this sense, novel fast marching methods are now capable of dealing with uncertainty, dynamic objects, and kinematics of the vehicle during the orientation process as we have shown in Pêtrès and Patrón (2005); Pêtrès et al. (2007, 2009).

1.2.3.5 Decision Making

Autonomous decision making in robotics is the ultimate goal for autonomy. This challenge is tackled by mission planning, which is the topic of this thesis. Decision making is a function of context, knowledge and reasoning. But also, and no less importantly, it relates to the level of human interaction. Increasing the adaptability of a platform provides independence from the operator. However, it is necessary to increase the trust of the operator in such capabilities for real acceptance to occur. Only under these circumstances, we have shown that delegation of the operator on this autonomy can be ultimately achieved (Johnson et al., 2007). At the moment, operators trust vehicles to follow waypoints but do not yet trust vehicles to make autonomous decisions. Therefore, it is still necessary for the operator to remain in the control loop, making mission level decisions.

In Figure 1.3 we see the relationship between operator and unmanned vehicle. The unmanned vehicle has three levels of control: basic low-level control, trajectory management and mission management.

The trajectory planning techniques described in the previous section have achieved autonomous adaptation for the control of the trajectory of the platform. They provide classical procedural waypoint-based interaction with the operator. At this level of abstraction, operator's trust is achieved through the reporting of the mission execution state from the trajectory manager.

However, autonomous decision making solutions must allow human interaction at higher levels of control. These solutions must be designed around human-centric

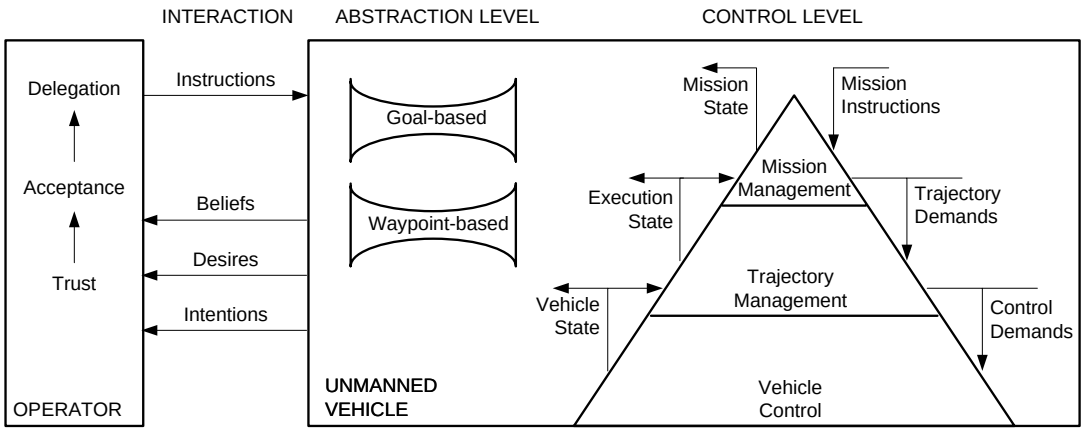


Figure 1.3: Levels of control of a robotic system (right), levels of abstraction of interaction with the operator (center), and process flow for delegation of decision making tasks to the autonomous system (left).

needs (Platts, 2006) that are easy to interpret by the operator. Thus, solutions that are able to operate with limited communication and that require reduced technical knowledge of the platform by the operator. At this level, the operator only requires knowledge about the beliefs, desires and intentions of the vehicle for trusting its autonomous actions (Miller, 2005). This can be achieved through the use of declarative goal-based mission planning approaches (Richards and Howitt, 2006).

Declarative models are static representations about events, objects and their relationships. Goal-based planning focuses on the 'what-to' side of the mission instead of the 'how-to'. At this level of abstraction, the operator is kept informed through the mission state report from the mission manager as shown in Figure 1.3.

The problem is that current mission plan managers for underwater platforms are procedural and static (Hagen, 2001). If behaviours are added (Pang et al., 2003), they are only to cope with possible changes that are known *a-priori* by the operator. In order to achieve true autonomous decision making, adaptability should be extended from current procedural waypoint-based approaches on trajectory planning to declarative goal-based solutions for mission planning.

The need for autonomous declarative goal-based solutions for adaptive mission planning is the main motivation behind this thesis.

1.3 Autonomous Decision Making

Autonomous decision making is the the ultimate goal for autonomy as described in the challenge above. In this thesis we attempt to solve this challenge by using adaptive mission planning. This section defines the goals, implications and environment of autonomous decision making. We highlight the contribution of adaptive mission planning to the improvement of UUVs operability, which is the overall goal of maritime research. We then describe scenarios which are intended to benefit from our research, which motivate our final approach. Finally, we state the overall objectives of this research.

1.3.1 Operability

Operability is a term which defines how accessible the platform is to the final user. It is at the core of improving effectiveness for robotic systems. Improving (↑) operability improves the availability of a platform, its affordability and its final acceptance (see Glossary of Terms for the definition of these terms):

$$\uparrow \text{Operability} \Rightarrow \uparrow \text{Availability} \wedge \uparrow \text{Affordability} \wedge \uparrow \text{Acceptance} \tag{1.1}$$

Two main characteristics can improve operability: reliability relates to the system failures due to the internal hardware components, and survivability relates to failures due to external factors or damages. This relationship is denoted in Equation 1.2.

$$\uparrow \text{Survivability} \vee \uparrow \text{Reliability} \Rightarrow \uparrow \text{Operability} \tag{1.2}$$

Thornton (2005) highlighted that the current emphasis for increasing operability is focused on increasing the survivability of the platform. This is achieved by decreasing (↓) its susceptibility and vulnerability (see Equation 1.3). Examples of this effort are trajectory planning techniques described in Section 1.2.3.4.

$$\uparrow \text{Trajectory Adaptability} \Rightarrow \downarrow \text{Susceptibility} \vee \downarrow \text{Vulnerability} \Rightarrow \uparrow \text{Survivability} \tag{1.3}$$

Although much research is currently aimed at making platforms more survivable, we focus on making platforms more reliable. When active and passive measures fail to protect the platform, or other kind of unexpected events occur, the focus of the mission should shift to 'reconfigure' itself, i.e. to use alternative combinations of remaining resources. Recoverability is a critical capability for the endurance of the platform (JRP

Master Plan, 2005). Recoverability can be provided by autonomous mission planning adaptation techniques (see Equation 1.4).

$$\uparrow \text{Mission Adaptability} \Rightarrow \uparrow \text{Recoverability} \Rightarrow \uparrow \text{Reliability} \tag{1.4}$$

1.3.2 Scenarios

This section describes a set of possible scenarios from the maritime disciplines described in Section 1.2.1 that would benefit from the increase in operability provided by on-board adaptive mission planning for autonomous decision making in UUVs. This section provides important background on the intended functionality and operational constraints of the final approach.

1.3.2.1 Autonomous Exploration and Discovery

Under-ice exploration in oceanography (Ackley et al., 2008), long-distance inspection of subsea structures for the energy industry (Patrón et al., 2006a; Evans et al., 2009), and surveillance of harbour structures and ship hulls for change detection for the Navy (UUV Master Plan, 2004) require autonomous decision making when communication links are missing.

Under these scenarios, it is expected that UUVs can be launched in areas with high levels of uncertainty to collect information over a predetermined period of time. These missions require high levels of autonomy without communicating with other intelligent agents, including the operator. In these environments, the vehicle is expected to adapt to changes in the environment or in the platform components that compromise or affect the current mission plan. This adaptation should be achieved based on the perception of the environment and previously defined mission objectives.

1.3.2.2 Reactive Data Gathering and Intervention

Additionally, autonomous biological sampling for oceanography (Curtin et al., 1993), smart offshore operations for the energy industry (Saul and Tena, 2007), and mine disposal for the Navy (Brown, 2003) require reactive data gathering and intervention capabilities.

Autonomous mission plan adaptation is required for reacting to and interacting with the environment. This adaptation should be based on incoming sensor data and initial user requirements. This maximises efficiency of water time and gathering of

useful data. Thus, reducing the overall mission cost. Mission plan adaptation for achieving full autonomous decision making will provide flexible, efficient, and cheaper data gathering and intervention capabilities than current solutions.

1.3.2.3 Subsea Sensing Networks for Long-term Deployment

Finally, ocean observatories in oceanography (Ocean Observatories Initiative, 2000), all year round subsea field inspection for the energy industry (Jamieson and Tena, 2009), and continuous harbour and coast patrolling for the Navy (US DoD RoadMap, 2009) are rapidly emerging. They require a more permanent presence of robotic and sensing tools underwater.

These applications demand underwater networks of fixed sensors in combination with fleets of AUVs and gliders. These sensing webs require decision making algorithms in order to optimise the management of heterogeneous assets and resources, to couple observation systems and ocean models, to minimise error in the reconstruction of ocean fields, and to provide fast dynamic response to events.

1.3.3 Research Objectives

Looking at the previously described scenarios, this section aims to translate the broad challenges into specific objectives for this research. The main research objectives of this thesis are:

1.3.3.1 Autonomous Decision Making

First of all, the main focus of this research is to investigate what mission planing approaches are most appropriate for providing on-board real time autonomous decision making on UUVs under different circumstances. The mission environment is assumed uncertain but not chaotic. Changes, when they occur, are progressive and slow.

Coping with tight resource constraints and hard deadlines shall provide autonomy for long periods of time. Thus, increasing the operability of the platform for long endurance missions with limited communications and lack of human interaction. During these long periods, the approach should react accordingly to new environmental and platform constraints with the efficiency required by the vehicle specifications and constraints of the mission. Furthermore, renewable and non-renewable resources are required to be adequately scheduled based on the mission objectives. Consequently, it is expected to improve vehicle operability and rates of mission success.

1.3.3.2 Conceptualisation of Raw Data

As discussed in Section 1.2.3.3, the sensors monitoring the variability of the environment and the algorithms processing the sensor data provide different levels of abstraction and certainty. Environment and platform conditions should be induced from the partial input of sensory information installed on the platform. This demands a solution that places these observations in context. We will study different approaches enabling the abstraction from raw sensor data to the conceptual information required by the high level decision making. Describing the mission environment conceptually makes the information easier to understand for the human operator, and this makes the system easier to use and more trustworthy.

1.3.3.3 Service-based Decision Making

There is currently a trend towards systems based on plug-and-play architectures, which are extensible and flexible. The concepts of operations described in the scenarios of Section 1.3.2 present systems composed of multi-disciplinary payloads and sensors. Each of these elements provide different services (see Glossary of Terms for the definition of Service-oriented Architecture). All these services are required to publish their capabilities and resource constraints in a discoverable environment.

We will be looking at the most suitable planning techniques to autonomously determine the mission plan of the platform using the knowledge about different capabilities published by the different services.

1.3.3.4 Robustness and Portability

The approach should continuously reassess the status of the platform and the environment. As a result, it should dynamically adapt to them. In order to achieve this, robustness of the solution is essential.

In consequence, a dynamic balance between the rigour of the deliberative planning and the rapid response of the reactive behaviour will be required. Increasing the search time for a plan solution may not be operationally acceptable. On the other hand, increasing the performance of the hardware resources of the vehicle may not be available. The algorithm should be able to understand the capability of the available resources. Then, it should provide a mission plan that fits the platform and mission constraints in soft real-time.

Finally, we will analyse ways to make the approach scalable and portable to different platforms in different domains.

1.3.3.5 Measurable Performance

Ultimately, the approach should increase the reliability of the platform. This was identified as a critical element for improving its operability (see Section 1.3.1). We will be looking at different methods able to provide a quantitative measurement of this increase in operability. We also investigate the importance of semantic information in evaluating autonomous decision making.

A system capable of performing as described is expected to be very demanding. As the technologies progress, these demands will be more easily achieved. The approach should be able to perform in the current existent technologies but also be scalable to the new up-coming technologies. Current limited processor speeds and working memory space are expected to be the main bottlenecks to coping with the real time requirements.

1.4 Adaptive Mission Planning

As described in the previous section the main goal of our research is to tackle the unsolved challenge of autonomous decision making for underwater robotics. We have done this by using adaptive mission planning techniques. A mission planning system is adaptive if it provides timely and effective responses to changes in directives or environment. We describe the decision making process currently used by UUV systems and we introduce the unmanned decision loop, where observations, orientations, decisions and actions (OODA) occur in a loop enabling adaptive mission planning. We discuss how related work has tackled this challenge. Finally, we motivate and present our implementation of adaptive decision making by placing it in context with related work.

In order to describe the unmanned decision loop, we need to start by modelling the mission environment. We will use these terms frequently from here on.

Definition 1.4.1 *A mission environment is defined by the tuple* $\Pi = (\Sigma, \Omega)$*, where:*

- Σ *is the mission domain model containing information about domain, i.e. the platform and the environment of execution, and*

- Ω *is the mission problem model containing information about the problem, i.e. mission status, requirements, and objectives.*

The set of all possible mission environments for a given domain is defined as the *domain space* (e.g., the domain space of the underwater domain). It is denoted by Θ. A mission environment Π is an element of one and only one Θ.

From this model, a mission plan π that tries to accomplish the mission objectives can be produced. However, this mission environment evolves over time t as new observations of the domain model Σ_t and the problem model Ω_t continuously modify it:

$$\Pi_t \leftarrow \Pi_{t-1} \cup \Sigma_t \cup \Omega_t \tag{1.5}$$

The decision making process to calculate a mission plan π_t for a given mission environment Π_t occurs in a cycle of observe-orient-decide-act. This process was termed by Boyd (1992) as the OODA-loop, and it was modelled on human behaviour. Inside this loop, the *Orientation* phase contains the previously acquired knowledge and initial understanding of the situation of the mission environment (Π_{t-1}). The *Observation* phase corresponds to new perceptions of the mission domain model (Σ_t) and the mission problem model (Ω_t) that modify the mission environment. The *Decision* component represents the level of comprehension and projection. This last stage is the central mechanism enabling adaptation before closing the loop with the *Action* stage.

Note that it is possible to make decisions by looking only at orientation inputs without making any use of observations. In this case, Eq. 1.5 becomes $\Pi_t \leftarrow \Pi_{t-1}$. In the same way, it is also possible to make decisions by looking only at the observation inputs without making use of available prior knowledge. In this case, Eq. 1.5 becomes $\Pi_t \leftarrow \Sigma_t \cup \Omega_t$.

In current UUVs implementations, the human operator constitutes the decision phase. See Figure 1.4 for a schematic representation of the control loop. When high

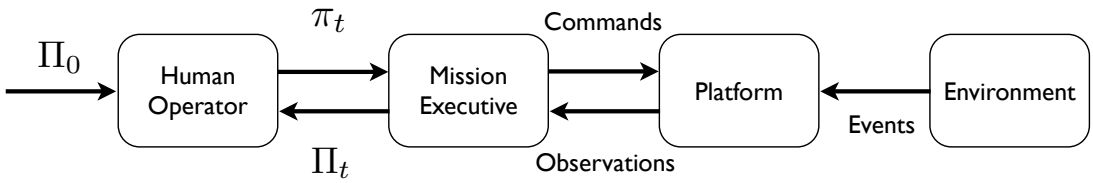


Figure 1.4: Observation, Orientation, Decision and Action (OODA) loop for unmanned vehicle systems with decision making provided by the human operator.

bandwidth communication links exist, the operator remains in the OODA-loop during the mission execution taking the decisions. For each update of the mission environment Π_t received, the operator decides on the correspondent mission plan π_t to be performed. From the list of actions in this mission plan, the mission executive issues the correspondent commands to the platform. Examples of the implementation of this architecture are existing Remotely Operated Vehicles (ROVs).

However, when communication is unreliable or unavailable, the operator must attempt to include all possible if-then-else cases to cope with execution alternatives before the mission starts. This is the case of current AUVs implementations that follow an orientation-only model. Figure 1.5 shows this model, where the OODA-loop is broken because observations are not reported to the human operator.

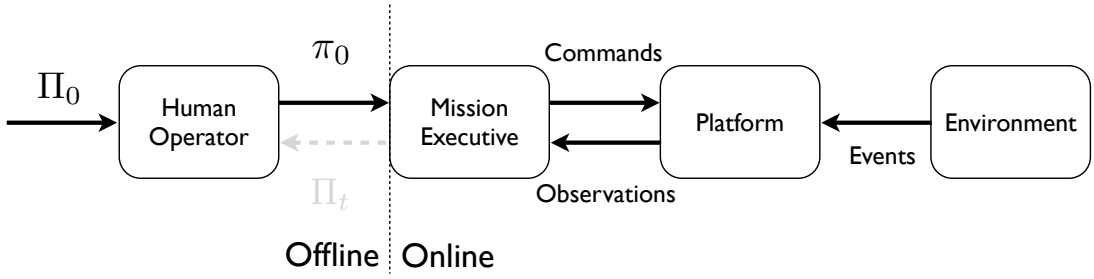


Figure 1.5: Broken OODA-loop. Decision stage on the human operator based only on initial pre-mission orientation.

We will now discuss a few recent AUV implementations which show where the state-of-the-art is currently positioned. Most implementations rely on pre-scripted mission plan managers that are procedural and static and might not even consider conditional executions (Hagen, 2001). At this level, the mission executive follows a sequence of basic command primitives and issues them to the functional control layer of the platform. Description about how these approaches maintain control of underwater vehicles can be found in Fossen (1994), Ridao et al. (1999) and Yuh (2000). In this situation, decisions taken by the operator are made using only orientation inputs related to some previous experience and *a-priori* knowledge. This has unpredictable consequences, in which unexpected situations can cause the mission to abort and might even cause the loss of the vehicle (Griffiths, 2005a; von Alt, 2010).

More modern approaches are able to mitigate this lack of adaptability by introducing sets of behaviours that are activated based on the observations perceived (Arkin, 1998). Behaviours divide the control system into a parallel set of competence-levels.

They can be seen as manually scripted plans generated *a-priori* to encapsulate the decision loop for an individual task. Under this approach, the key factor is to find the right method for coordinating these competing behaviours.

The subsumption model, attributed to Brooks (1986), arbitrates behaviour priorities through the use of inhibition (one signal inhibits another) and suppression (one signal replaces other) networks. Most recent AUV control systems are a variant of the behaviour-based subsumption architecture.

This model was first applied to the control of UUVs by Turner (1995) during the development of the ORCA system. This system used a set of schemas in a case-based framework. However, its scalability remains unclear as trials for its validation were not conducted.

Later, Oliveira et al. (1998) developed and deployed the CORAL system based on Petri nets. The system was in charge of activating the vehicle primitives needed to carry out the mission. These primitives were chained by preconditions and effects.

The scaling problem was addressed by Bennet and Leonard (2000) using a layered control architecture. Layered control is a variant of the subsumption model that restricts of interaction between layers in order to keep it simple (Bellingham et al., 1990). The system was deployed for the application of adaptive feature mapping.

Another approach for coordinating behaviours is vector summation that averages the action between multiple behaviours. Following this principle, the DAMN system developed by Rosenblatt et al. (2002) used a voting-based coordination mechanism for arbitration implementing utility fusion with fuzzy logic.

The MOOS architecture developed by Newman (2002) was also able to guide UUVs by using a mission control system called Helm. Helm's mission plan was described by a set of prioritised primitive tasks. The most suitable action was selected using a set of prioritised mission goals. It used a state-machine for execution, a simplified version of a Petri net.

The O^2CA^2 system (Carreras et al., 2007) also used a Petri net representation of the mission plan (Palomeras et al., 2006, 2009). The system maintained the low level control (dynamics) from the guidance control (kinematics) uncoupled (Caccia and Veruggio, 2000). Although it contained a declarative mission representation, missions were programmed manually.

A detailed survey of other behaviour-based approaches applied to mission control systems for UUVs can be found in Carreras et al. (2006).

More recently, Benjamin et al. (2009) has applied multiple objective decision the-

ory to provide a suitable framework for formulating behaviour-based controllers that generate Pareto-optimal and satisfying behaviours. This approach was motivated by the infeasibility of optimal behaviour selection for real-world applications. This approach has been implemented and deployed as part of the IvP Helm extension to MOOS. This method seems to be a more suitable for behaviour selection, although more computationally expensive. Also, the approach is limited to the control of only the direction and velocity parameters of the host platform.

After reviewing this related work, two problems affecting the effectiveness of the decision loop become clear. Firstly, orientation and observation should be linked together because it is desirable to place the new observations in context. Secondly, decision and action should be iterating continuously. These two problems have not been addressed together by previous approaches. These are two of the goals that we address in this thesis. In order to achieve them autonomously, two additional components are required: a status monitor and a mission plan adapter. The status monitor reports any changes detected in the mission environment during the execution of a mission. When the mission executive is unable to handle the changes detected by the status monitor, the mission planner is called to generate a new modified mission plan that agrees with the updated mission environment. Figure 1.6 shows the OODA-loop for autonomous decision making. Comparing it to the previous Figure 1.5, the addition of status monitor and mission planner removes the need for human decisions in the loop. Note that the original mission plan π_0 could also be autonomously generated as long as the high-level goals are provided by the human operator in Π_0.

Adaptive mission planning enables a true unmanned OODA-loop. This autonomous decision making loop copes with condition changes in the mission environment during the mission execution. As a consequence, it releases the operator from decision making tasks in stressful environments containing high levels of uncertainty and dynamism.

The potential benefits of adaptive mission planning capabilities for autonomous decision making in UUVs were promoted by Turner (2005), Bellingham et al. (2006) and Patrón and Petillot (2008). Possibly the most advanced autonomous decision making framework for UUVs has been developed at the Monterey Bay Aquarium Research Institute. This architecture, known as *T-REX*, has been deployed successfully inside the Dorado AUV (Rajan et al., 2009). This is now providing adaptive planning capabilities to oceanographers for maximising the science return of their AUV missions (Rajan et al., 2007; McGann et al., 2007). Using deliberative reactors for the concurrent integration of execution and planning (McGann et al., 2008a), live sensor data can

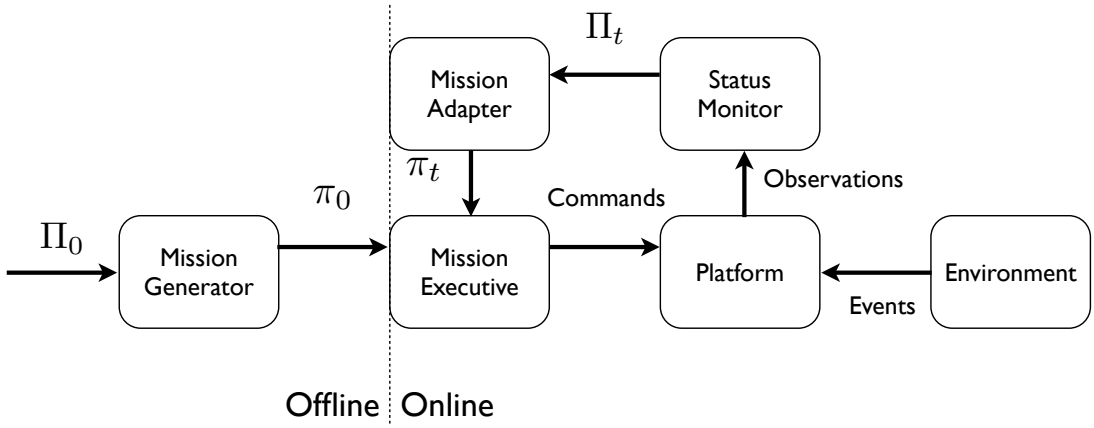


Figure 1.6: Required OODA-loop for autonomous decision making in UUVs. Decision stage for adaptation takes place on-board based on initial orientation provided by the operator and observations provided by the status monitor.

be analysed during mission to adapt the control of the platform in order to measure dynamic and episodic phenomenon, such as chemical plumes (McGann et al., 2008b, 2009). Alternative approaches to adaptive plume tracing can also be found in the works of Farrell et al. (2005) and Jakuba (2007). Their research goals of all these approaches have been motivated by scientific applications and do not consider the needs of the human operators or the maritime industry.

However, autonomy cannot be achieved without humans, as it is necessary for this autonomy to be ultimately accepted by an operator. Our research is geared towards improving human access to UUVs in order to solve the maritime industry's primary requirement of improving platform operability (Patrón et al., 2007a). We propose a goal-based approach to solving adaptive mission planning. The advantage of this approach is that it provides high levels of mission abstraction. This makes the human interface simple, powerful and platform independent, which greatly eases the operator's task of designing and deploying missions (Patrón, 2009). Ultimately, operators will not need any specialist training for an UUV specific platform, and instead missions will be described purely in terms of their goals. Apart from ease of use, we have also demonstrated using a novel metric (Patrón and Birch, 2009) that adaptive mission planners can produce solutions which are close to what a human planner would produce (Patrón et al., 2009a). This means that our solutions can be trusted by an operator.

Another advantage of our research over other state-of-the-art UUV implementa-

tions, is that we are industry focussed. Our service-oriented approach provides goal-based mission planning with discoverable capabilities, which meets industry's need for platform independence (Patrón et al., 2009b). Finally, our plan repair approach optimises the resources required for adaptability and maximises consistency with the original plan, which improves human acceptance of autonomy. Resource optimisation and consistency are very important properties for real world implementations, as we demonstrate in our sea trials (Patrón et al., 2008b).

So our system goes beyond previous research along two dimensions. It is extending the availability and acceptance of UUVs, through demonstrating improvements in operability, as stated in Equation 1.1.

1.5 Research Contribution

Our research is geared towards improving human acceptance of higher levels of vehicle autonomy while at the same time addressing the maritime industry's main goal of improving platform operability. This thesis applies the most appropriate tools provided by the latest planning research to the underwater domain, where low levels of autonomy are currently the norm.

In meeting the objectives described in Section 1.3.3 this research combines: three goal-based approaches for different scenarios requiring adaptive mission planning; a semantic-based framework to integrate orientation and observation; and metrics for evaluating the results of the different adaptive strategies.

These novel contributions are detailed in the following list below:

1. First, in this chapter, we identify the elements affecting the effectiveness of the decision making loop. We justify how orientation and observation should be linked together and decision and action should iterate continuously in order to allow autonomous decision making for UUVs.

2. We identify unmanned situation awareness as necessary for autonomous decision making and that this is unavailable in current UUVs platforms. We develop a novel semantic-based knowledge representation framework integrating initial expert orientation and the observations acquired during mission. This framework provides data conceptualisation, semantic knowledge interoperability, and reasoning among all information sources involved during a mission. We demonstrate how this framework can act as an enabler for autonomy and on-board de-

cision making. It provides higher-level interaction with the operator, liberating them from the low-level understanding of the platform functionalities.

3. We analyse current metrics measuring the adaptation process of a mission plan. Then, we introduce two novel metrics. Plan Proximity compares planning strategies by measuring the difference between the action sequence of the plans and the expected outcome states. We show that we can quantify the amount of change introduced by adaptive mission planning strategies. Semantic Plan Proximity extends the weights and mechanisms in the formulation of the metric in order to contain semantic information. By making use of some valid assumptions, we are able to maintain the semantic metric independent from any domain-specific measurements.

4. After analysing current approaches to adaptive mission planning, we present a novel continuous mission planning approach specifically suited to UUVs operating in dynamic and uncertain discoverable mission environment. The approach implements prediction, measurement and correction inside a Markov decision process framework. We implement a policy that balances the selection of plans by using their estimated cost of execution and the reward obtained by reaching the new configuration of the environment. We are able to forecast the impact of sensed events in the precomputed plan and, if necessary, react in advance. This approach handles temporal planning with durative actions, metric planning, opportunistic planning and dynamic planning. Its evaluation under a static scenario and a partially-known dynamic scenario using Plan Proximity as a metric provides similar results as the humanly driven mission.

5. We analyse mission planning under a service-oriented architecture. We identify that the total set of actions of a system is the union of all the functionalities provided by the services in the system. We implement a goal-based approach based on backward state-space search that makes use of service discovery capabilities to solve the selection of state candidates during the search process. This approach provides platform independence and eases the creation of mission plans. We demonstrate that the discovery-based implementation finds the same results as the baseline which is explicitly provided with the platform configuration.

6. We identify how platforms have limited computational resources to perform the consuming task of generating a mission plan. We compare plan regeneration

and plan repair and find that plan repair is less resource intensive while also producing plans that are more proximate to the original. Our mission plan repair approach is based on a partial mission plan representation using an iteration of unrefinement and refinement stages. We provide this mission plan repair at planning and executive levels. Finally, we deploy an implementation of a semantic-based autonomous planning system in a real vehicle. The system was shown capable of providing embedded autonomous mission plan adaptation for maintaining operability of a host UUV platform in the face of unexpected events.

1.6 Summary and Outlook

This chapter presented the research challenges, objectives, and contributions of this thesis. First, it identified unmanned underwater vehicles as the key technology for the improvement of maritime capabilities in hazardous and uncertain environments. From all the challenges that UUVs are facing in the underwater domain, adaptive mission planning was identified as the most direct way to increase recoverability and reliability and hence operability. The research objectives were extracted by looking at a set of underwater scenarios. After analysing current approaches tackling these objectives, this chapter proposed a new autonomous decision making framework for underwater robots that is based on the human loop of orientation, observation, decision and action (OODA). While discussing related work, we identify the novelty of our approach and detail our contributions.

The rest of the thesis is divided into chapters with their dependencies shown in Figure 1.7. Each chapter largely focuses in one of the research objectives stated in Section 1.3.3 and follows a similar template: First, an introduction describes the focus of the chapter. Then, a detailed literature review is presented, describing current approaches to the objectives of the chapter. This is followed by a description of our approach to solving the objectives of the chapter. Finally, a validation, evaluation or illustration of the approach is described. Each chapter finishes with a summary and a list of key related publications.

The chapters in this thesis are: Chapter 2 defines a novel semantic world model framework for hierarchical distributed representation of knowledge in autonomous systems. In Chapter 3, Plan Proximity is proposed as a distance-based metric for the evaluation of adaptive mission planning approaches. Chapter 4 proposes a novel approach for adaptive mission planning for UUVs operating in dynamic and uncertain

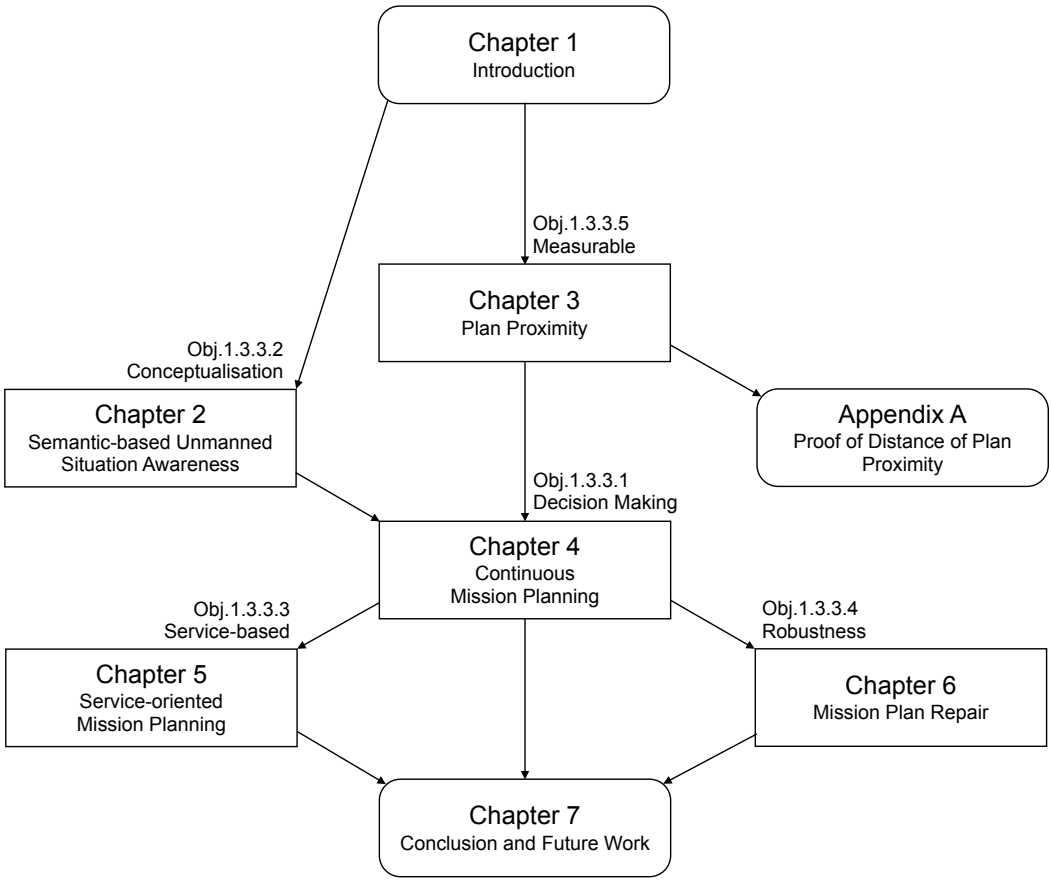


Figure 1.7: Dependencies among the chapters. A line means that one chapter is needed in order to understand another. Boxes with square corners represent chapters solving the research objective displayed above them.

discoverable environment. Chapter 5 combines the benefits of the knowledge-based framework and goal-based mission planning in order to provide interoperability of embedded intelligent agents for distributed service discovery and embedded decision making. Chapter 6 implements and deploys a semantic-based autonomous planning system capable of providing embedded autonomous mission repair at planning and execution levels for maintaining operability of UUVs in the face of unexpected events. Chapter 7 summarises the results and achievements obtained in this piece of research and proposes the next steps to be taken in the future.

Finally, Appendix A demonstrates the distance properties of the Plan Proximity metric and Appendix B illustrates an example of a description of a mission environment scenario. The thesis concludes with the Bibliography section that contains a list of the articles referenced.

1.7 Key Publications

- The need of adaptive mission planning in underwater vehicles was highlighted in the editorial's soapbox *Embedded knowledge and autonomous planning* published in the **Sea Technology Magazine** on April 2009 (Patrón, 2009).
- The underwater environment challenges described in Section 1.2.3 were included in *The underwater environment: A challenge for planning* paper presented at the **UK PlanSIG Workshop** on December 2008 (Patrón and Petillot, 2008).
- The perception limitations from Section 1.2.3 were included in the paper *Strategies and Sensors Technologies for UUV Collision, Obstacle Avoidance and Escape* presented at the 7^{th} **Unmanned Underwater Vehicle Showcase** (UUVS'05) on September 2005 (Patrón et al., 2005) and in an article of the **Journal of the Undersea Defence Technology Forum** on 2007 (Patrón et al., 2007b).
- The paper *Path Planning for Unmanned Underwater Vehicles* presented at the **Workshop on Planning and Learning in A Priori Unknown or Dynamic Domains of the** 19^{th} **International Joint Conference on Artificial Intelligence** (IJCAI'05) on August 2005 (Pêtrès and Patrón, 2005) contained the trajectory planning approach of Section 1.2.3.4. The extension to directional and curvature constraints was presented in *Path Planning for Autonomous Underwater Vehicles* of the **Journal of IEEE Transactions on Robotics** issue of April 2007 (Pêtrès et al., 2007) and in the Chapter *Trajectory Planning for Autonomous Underwater Vehicles* of the **Book of Underwater Vehicles** (Pêtrès et al., 2009).
- The management of navigation uncertainties presented in Section 1.2.3.4 appeared in *A smooth simultaneous localisation and mapping solution to improve situational awareness, mission planning and re-planning for AUVs* at the **World Maritime Technology Conference - Advances in Technology for Underwater Vehicles** (WMTC'06) on March 2006 (Patrón and Tena-Ruiz, 2006). Its evaluation was summarised in *Design and Evaluation of a Reactive and Deliberative Collision Avoidance and Escape Architecture For Autonomous Robots* of the **Journal of Autonomous Robots** on December 2007 (Evans et al., 2008).
- The need of delegation from Section 1.2.3.5 appeared in *The importance of trust between operator and AUV: Crossing the human/computer language barrier* at the **IEEE Oceans Europe Conference** on June 2007 (Johnson et al., 2007).

Chapter 2

Semantic-based Unmanned Situation Awareness

The eye sees only what the mind is prepared to comprehend.

– Henri Bergson

2.1 Introduction

In the introduction, we motivated the need for adaptive mission planning. We described how this was possible within an unmanned decision loop, where observations, orientations, decisions and actions (OODA) create an operational cycle. This chapter proposes a solution for integrating orientation and observation. Integrating these two elements allow the mission environment model to remain updated, and thus achieving situation awareness. As previously described, orientation refers to the current knowledge available to a mission and it comprises of both the initial understanding of mission environment, and knowledge previously acquired during the mission. Observation corresponds to the new perceptions acquired at the current time by the mission.

In this chapter we analyse current approaches and present a novel semantic world model framework that allows the integration of orientation and observation. Using a hierarchical distributed representation of knowledge, this framework provides semantic knowledge interoperability among all information sources involved in a mission. This allows different embedded agents to communicate while at the same time remaining independent of each other, responsible only for the services that they are providing. These services can also contribute to the world model, enriching the knowledge

available to all the agents. Another strength of our framework is that it can make use of heterogeneous real-world data of very different types. This data is processed by several different layers and agents, and is finally made available in a suitable format to high-level decision making agents. This allows us to abstract away from the raw real-world data in a step by step fashion, leveraging the power of semantic technologies. Our approach liberates the operator from needing a low-level understanding of the platform functionalities. Therefore, it makes the system easier to use and more trustworthy.

We illustrate the benefits of our approach using a scenario where the status monitor and the mission planner agents collaborate. The semantic-based framework is able to provide these agents with the required situation awareness for autonomous decision making. We also show that interaction with the human operator could be made easier, requiring the definition of high-level mission objectives rather than low-level platform commands.

2.2 Situation Awareness for Autonomy

At the human level, situation awareness is the real-world changing knowledge that is critical for effective decision making before action. The human capability of understanding highly dynamic and complex environments is known as *situation awareness* (SA_H). SA_H breaks down into perception of the environment, comprehension of the situation and projection of the future status (Endsley et al., 2003).

Situation awareness in humans (SA_H) is limited by several factors:

- At the perception or observation level, the limit of how many elements one can perceive and pay attention to at one time affects how much information can be processed.
- During the comprehension phase, *working memory* is used for combining new perception with existing knowledge. Human working memory is very limited. Thus, it forms a bottleneck for SA_H.
- For the projection phase, *mental models* coming from the orientation phase are used. Mental models are based on both semantic knowledge and system knowledge. Semantic knowledge is knowing what, as opposed to how. System knowledge is understanding how the system works. This knowledge is very difficult to acquire and requires a large number of hours of training.

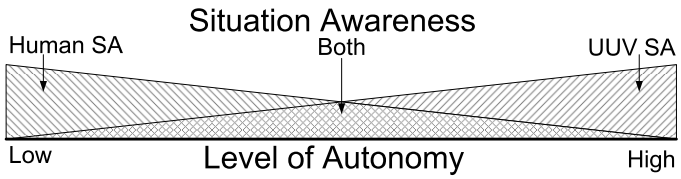


Figure 2.1: Human and vehicle situation awareness across the levels of autonomy showing the need for an increase in unmanned vehicle situation awareness in order to enable higher levels of autonomy.

Current ROV and AUV implementations rely on the operator remaining situated while taking decisions. As a consequence, these factors lead to errors in the decision making process.

By transferring situation awareness from humans to machines most of these limitations can be removed. SA_H definitions can be directly applied to the notion of *unmanned vehicle situation awareness* (SA_V) (Adams, 2007). Increasing the levels of situation awareness for unmanned vehicle systems is a requirement for transferring from current full human control to fully autonomous unmanned capabilities (JRP Master Plan, 2005) (see Figure 2.1). Situation-aware systems are the paradigm for enabling autonomous decision making to solve problems for real-world scenarios (Wilkins and desJardins, 2001).

At present in underwater robotic platforms, knowledge representation is embryonic, applications are mono-domain, and mission objectives are mono-platform (Hagen, 2001). This limits the potential of multiple coordinated actions between agents. Consequently, the main application for UUVs is information gathering from sensor data. In a standard mission flow, data is collected during a mission and then post-processed off-line.

However, in order to be able to let decision making technologies evolve towards providing higher levels of autonomy and control, embedded service-oriented agents require access to higher levels of knowledge representation or abstraction.These higher levels are required to provide knowledge representation for contextual awareness, temporal awareness and behavioural awareness.

Two sources can provide this type of information: the original domain knowledge extracted from the expert (orientation) and the knowledge derived from the processed sensor data (observation) (see Section 1.4). In both cases it will be necessary for the information to be stored, accessed, and shared efficiently by the deliberative agents

while performing a mission. These agents, providing different capabilities, might even be distributed among the different platforms working in collaboration.

2.3 Towards Unmanned Situation Awareness

This section starts presenting related work in approaches solving the unmanned situation awareness problem for unmanned underwater vehicles. It introduces a list of current approaches highlighting the flaws encountered when used to represent the knowledge of the mission environment in an autonomous system. The second section introduces the proposed semantic world model framework and its components.

2.3.1 From Observations to the Full Picture

Knowledge representation approaches implemented in current world models are only able to provide the perception or observation level of SA_V. State of the art embedded agents make use of different message transfer protocols in order to maintain their vision of the mission environment updated. Several approaches can be found in the literature implementing information transfer protocols for robotics.

For example, robotic libraries such as Player (Collett et al., 2005) and Yet Another Robotic Platform (YARP) (Fitzpatrick et al., 2007), support transmission of data across various protocols – TCP, UDP, MCAST (multi-cast), shared memory. These libraries provide an interface to the sensors and actuators of the robot. This allows agents to read data from sensors, to write commands to actuators, and to configure devices on the fly. These two approaches separate the agents from the details of the network technology used but they do not provide any standardisation about the meaning of the information transferred.

Another attempt to provide orchestration of multiple robotic services to achieve complex behaviours is the Microsoft Robotic Developer Studio (2009) (MRDS). It provides concurrent library implementation for managing asynchronous, parallel tasks using message-passing and decentralized software services. It is proprietary software, and also operating system and computer language dependent.

On the other hand, the Robotic Operative System (ROS) (Willow Garage ROS, 2009) is open source. It includes hardware abstraction by implementing several different styles of communication, including synchronous RPC-style communication, asynchronous streaming of data, and storage of data. Like MRDS, ROS efforts tend to be

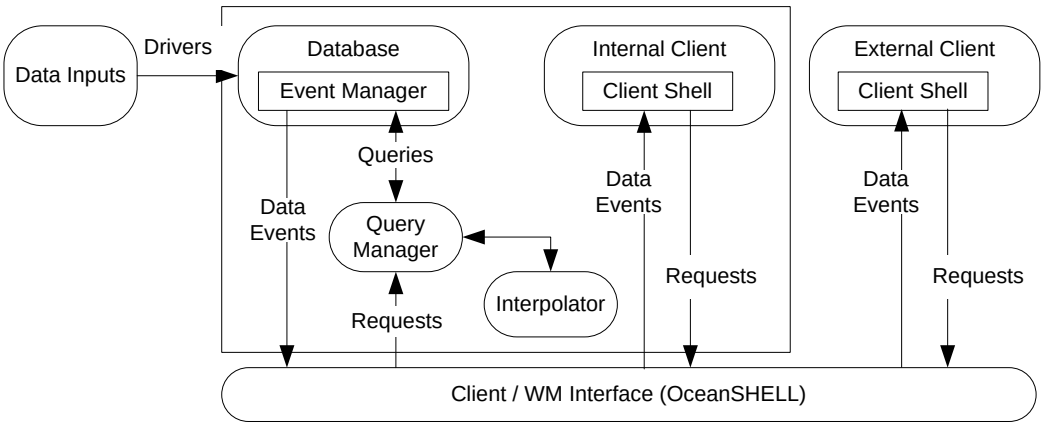


Figure 2.2: World model architecture for the BAUUV MoD program (Battlespace access for unmanned underwater systems). Its database handles data inputs and queries from internal and external clients.

directed towards indoor ground robotics. Additionally, this approach is limited to the type of network protocols that can be used and it does not standardise the semantics of the information transferred between agents.

The Mission Oriented Operating Suite (MOOS) (Newman, 2009a,b) uses human readable ASCII messages for communication of data to a centralised database. The database is simply a blackboard; an entity which stores the current state of the system. This centralised topology is vulnerable to 'bottle-necking' at the server as the ASCII messages can generate considerable parsing overheads.

The OceanSHELL libraries (OceanSHELL, 2005) implement UDP broadcast protocol to pass data information between agents. In the latest version, abstraction messages are able to carry semantic information. These libraries are the first attempt to standardise semantic information for unmanned underwater platforms in order to share knowledge between different multidisciplinary agents. However, the approach is still limited to the observation level of SA_V.

A more detailed review of these and other robotic knowledge representation approaches was published by Somby (2007).

As an extension of the OceanSHELL libraries, the Battlespace Access for Unmanned Underwater Vehicles (BAUUV) program, sponsored by the UK Ministry of Defence Directorate of Equipment Capability (Underwater Battlespace), showed the benefit of also including the mental or orientation model component of SA_V (Arredondo et al., 2006). A diagram describing the BAUUV dynamic multi-layered world model

architecture is shown in Figure 2.2. The approach was capable of integrating the capabilities of different agents by providing them with a common picture of the domain. This picture contained processed sensor data and *a priori* knowledge. Thus, it provided a full SA_V. However, the design was limited to detection and classification applications providing Autonomous Target Recognition (ATR) capabilities for the underwater domain. Therefore, the internal clients for information fusion were specific to the external agents requirements. In order to provide a truly service-oriented architecture, an evolution from this approach is required providing a generic framework independent of service capability and domain of application.

The Joint Architecture for Unmanned Systems (JAUS), originally developed for the Unmanned Ground Vehicles (UGVs) domain only, has recently extended to all other domains, i.e. air and water. It is trying to provide a common set of architecture elements and concepts (SAE-AS-4 AIR5665, 2008). Inside these different architectural elements, the JAUS model separates the service-oriented agents, called Functional Agents, into six different functional sets: Command, Tele-communications, Mobility, Payload, Maintenance and Training. In order to handle the orientation and observation phases, it classifies four different sets of Knowledge Stores: Status, World map, Library and Log.

Our experience has shown that overlap exists between these different sets of knowledge stores. This overlap is provided by the strong interconnection existing between the orientation and observation phases for SA_V. We propose an approach that makes use of some of the concepts proposed by JAUS but enhances the Knowledge Store set by providing higher flexibility in the way the information can be handled and accessed.

2.3.2 Semantic-based Unmanned Situation Awareness

The SA_V enables the vehicle to autonomously understand the '*big picture*'. As discussed, this picture is composed of the experience gained from previous experience (orientation) and the information obtained from the sensors while on mission (observation).

The approach proposed builds this picture with ontologies. Ontologies allow the representation of knowledge of these two components. Ontologies are models of entities and interactions, either generically or in some particular area of knowledge. Gruber (1995) defines an ontology as 'the specification of conceptualisations'.

The main components of an ontology are concepts and axioms. A concept repre-

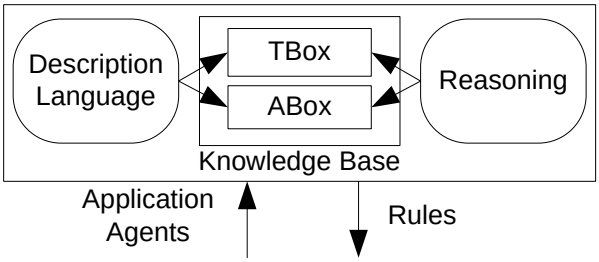


Figure 2.3: Knowledge Base representation system including the $TBox$, $ABox$, the description language and the reasoning components. Its interface is made of orientation rules and agent queries.

sents a set or class of entities within an application (e.g., a *fault* is a concept within the application of diagnostics). Axioms are used to constrain the range and scope of the concepts (e.g., a *driver* is a *software* that has a *hardware*). The finite set of concept and axiom definitions is called the 'terminological component' of the ontology, or $TBox$.

Instances are the individual entities represented by a concept of the ontology (e.g., a *remus* is an instance of the concept *AUV*). Relations are used to describe the interactions between individuals (e.g., the relation *isComponentOf* might link the individual *SensorX* to the individual *PlatformY*). This finite set of instances and relations about individuals is called the 'assertion component' of the ontology, or *ABox*.

The combination of a $TBox$ and a *ABox* is known as a *Knowledge Base* (see Figure 2.3). It can be seen how the $TBox$ and *ABox* align naturally to the orientation and observation components of SA_V respectively. Hence, the direct application of ontologies to the problem of unmanned situation awareness.

In the past, authors such as Matheus et al. (2003) and Kokar et al. (2009) have used ontologies for situation awareness in order to assist humans during information fusion and situation analysis processes. Our work extends these previous works by using ontologies for providing unmanned situation awareness in order to assist autonomous decision making algorithms in underwater vehicles. One of the main advantages of using a knowledge base over a classical data base schema to represent SA_V is the extended querying that it provides, even across heterogeneous data systems. The metaknowledge within an ontology can assist an intelligent agent (e.g., status monitor, mission planner, etc.) with processing a query. Part of this intelligent processing is due to the capability of reasoning. This enables the publication of machine understandable meta-data, opening opportunities for automated information processing and analysis.

For instance, a status monitor agent using meta-data about sensor location could

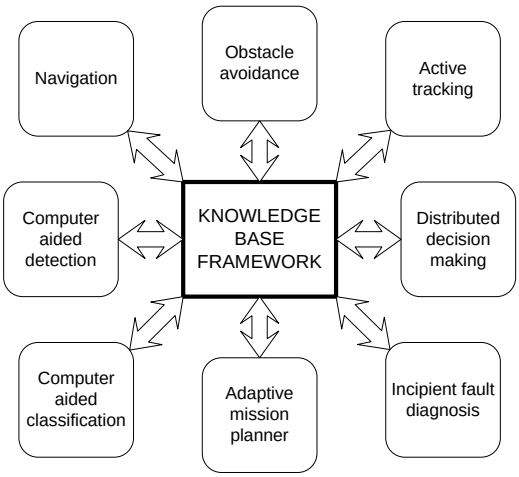


Figure 2.4: The knowledge base framework provides a common representation of knowledge for interoperability between multidisciplinary embedded agents.

automatically infer the location of an event based on observations from nearby sensors (Miguelanez et al., 2008). Inferences over the ontology are made by reasoners. A reasoner enables the domain logic to be specified with respect to the context model and applied to the corresponding knowledge, i.e. the instances of the model. A detailed description of how a reasoner works is outside of the scope of this thesis. For the implementation of the approach, the open source reasoner Pellet has been used (Sirin et al., 2007).

In this chapter, a novel knowledge framework for building a full SA_V is proposed. We have presented this framework in Patrón et al. (2008a) and Miguelanez et al. (2010). This framework is made of a library of knowledge bases. It provides a common machine understanding representation of knowledge between embedded agents that is generic and extendable (see Fig 2.4). It also includes a reasoning interface for inferring new knowledge from the observed data and knowledge stability by checking for inconsistencies. It improves local (service level) and global (system level) unmanned situation awareness. Thus, it can act as an enabler for autonomy and on-board decision making.

In order to provide a design that supports maximum re-usability (van Heijst et al., 1996), a three-level segmentation structure is adopted that includes the (1) Foundation, (2) Core and (3) Application ontology levels (see Figure 2.5).

Foundational Ontologies (FOs) represent the very basic principles and include Upper and Utility Ontologies. Upper ontologies describe generic concepts (e.g., the Suggested Upper Merged Ontology or SUMO (Niles and Pease, 2001)) while Util-

ity ontologies describe support concepts or properties (e.g. OGC_GML for describing geospatial information (Portele, 2007)). FOs meet the requirement that a model should have as much generality as possible, to ensure re-usability across different domains.

The Core Ontology provides a global and extensible model into which data originating from distinct sources can be mapped and integrated. This layer provides a single knowledge base for cross-application agents and services (e.g., vehicle resource / capabilities discovery, vehicle physical breakdown, and vehicle status). A single model avoids the inevitable combinatorial explosion and application complexities that results from pair-wise mappings between individual meta-data formats and ontologies.

In the bottom layer, an Application Ontology provides representation of knowledge of the particular expertise of each of the embedded agents.

Figure 2.6 represents the relationship between the Foundation Ontologies (Upper and Utility), the Core Ontology, and the Application Ontology for each service-oriented agent. Raw data gets parsed from sensors into assertions during the mission using a series of adapter modules for each of the sensing capabilities. It also shows that the knowledge handling of the agent during its decision making process is aided by the reasoner and the rule engine process.

2.3.2.1 Foundation and Core Ontology

Several institutions and consortium are currently developing standards for describing generic ontologies for the domains of unmanned platforms (SAE-AS-4 AIR5665, 2008) and underwater environments (Marine Metadata Interoperability, 2008).

To lay the foundation for the knowledge representation of unmanned vehicles, consideration was placed on the concepts described by Joint Architecture for Unmanned Systems (JAUS) domain model (SAE-AS4 AIR5664, 2006). Within the proposed framework, JAUS concepts are considered as the Upper Ontology for the knowledge representation. The Core Ontology developed in this work extends these concepts

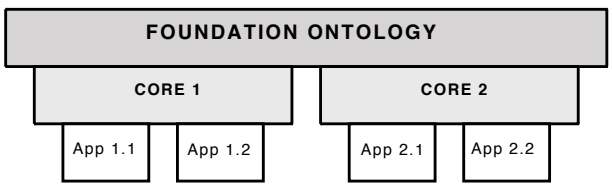


Figure 2.5: Levels of generality of the library of knowledge bases for SA_V. They include the Foundation Ontology, the Core Ontology, and the Application Ontology levels.

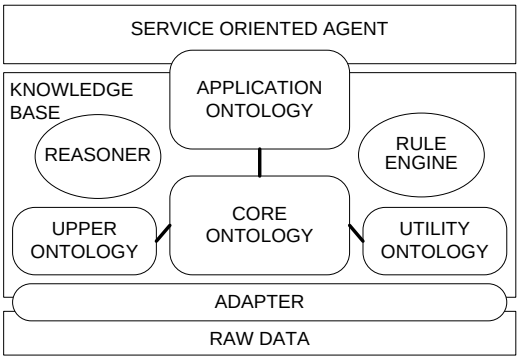


Figure 2.6: SA_V representation in the Knowledge Base using Core and Application ontologies supported by Upper and Utility ontologies. Generation of instances from raw data is performed by the Adapter. Handling of knowledge is done by the Reasoner, Rule Engine and the Service-Oriented Agent.

while remaining focused in the domain of unmanned systems.

Some of the Core Ontology concepts identified are:

- Platform: Static or mobile (ground, air, underwater vehicles),
- Payload: Hardware with particular properties, sensors or modules,
- Agent: Software with specific capabilities,
- Sensor: A device that receives and responds to a signal or stimulus,
- Driver: Module for interaction with a specific sensor / actuator,
- Waypoint: Position in space with coordinate and tolerance,
- Coordinate: Local frame, global frame, angular.

Figure 2.7 shows a snapshot of the Core Ontology representing the relations around the concept 'Platform'.

2.3.2.2 Application Ontology

Each service-oriented agent has its own Application Ontology. They represent the situation awareness of each of the individual agents. They include concepts that are specific to the expertise or service provided by the agent.

Embedded agents do not only use sensor information. They may also require information produced by another agents. A set of possible agent pairs of producer-consumer

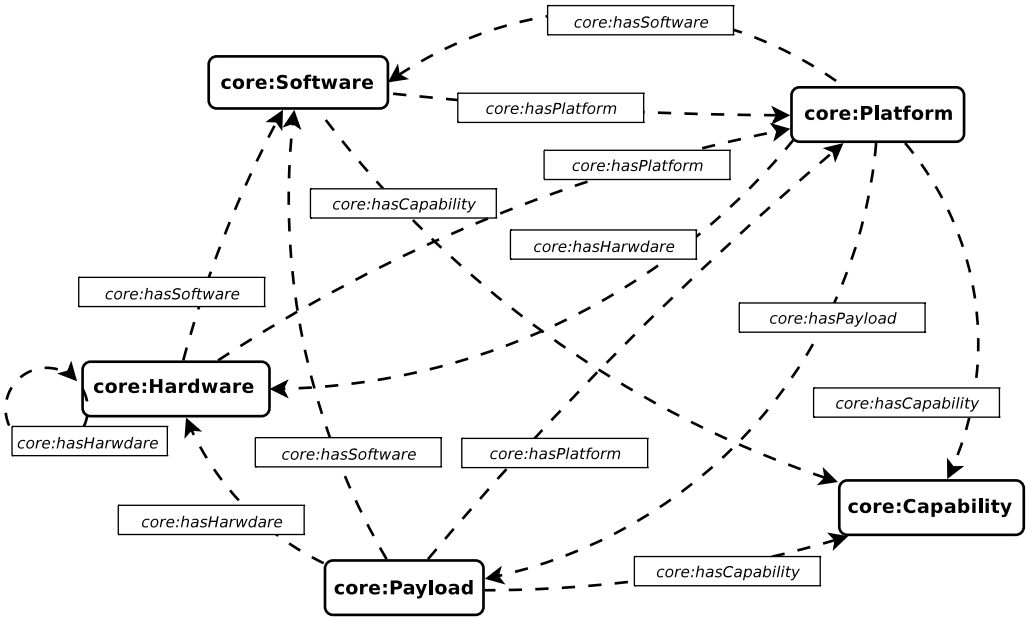


Figure 2.7: Snapshot of the Core Ontology representing the TBox environment – concepts (boxes) and axioms (arrows) – around the concept 'Platform'.

of knowledge are represented in Figure 2.8: On the top left, the pair Navigation agent and Obstacle avoidance agent require sharing information for geolocating and orienteering the platform (see Section 1.2.3.4). On the bottom left, the pair of Detection and Classification agents provide Autonomous Target Recognition capabilities, as described in Section 2.3.1. The framework can be extended to provide a common picture to decision making agents located in distributed platforms (see top right of the figure). This capability is presented as an extension in the future work suggested in Section 7.3.1. The case study presented in the following section concentrates the interaction and transfer of information between the pair formed by the status monitor agent and the mission planner agent (see bottom right of the figure).

2.4 The Monitor-Planning Scenario

This section focuses on the interaction between the status monitor and the mission adapter agents. The interaction between these agents has been described in Section 1.4. This interaction is the requirement needed to close the OODA-loop in an autonomous manner (see Figure 1.6). It enables autonomous on-board decision making for the adaptation of mission plans.

The next sections describe the correspondent Application Ontologies of the two

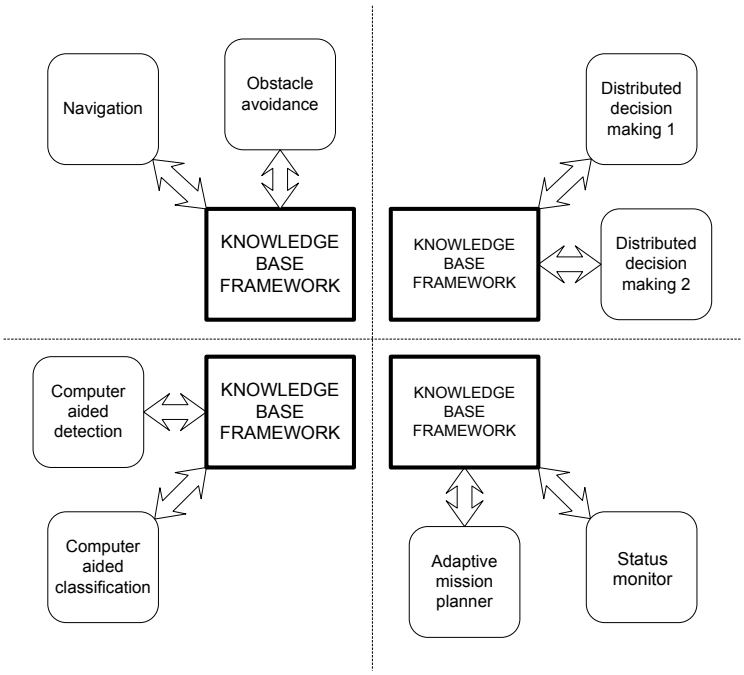


Figure 2.8: Possible scenarios pairs of agents for the production and consumption of knowledge.

agents.

2.4.1 Status Monitoring Application Ontology

The Status Monitoring Application Ontology is used to express the SA_V of the status monitor agent. The Status Monitoring Application Ontology is designed and built based on ontology design patterns (Blomqvist and Sandkuhl, 2005). Ontology patterns promote re-usability and consistency between domains. In this work, the representation of the monitoring concepts are based on the system-observation pattern represented in Figure 2.9. Some of the most important concepts identified for status monitoring are:

- Data: all internal and external variables (gain levels, water current speed),
- Observation: patterns of data (sequences, outliers, residuals,...),
- Symptom: individuals related to interesting patterns of observations (e.g., low gain levels, high average speed),

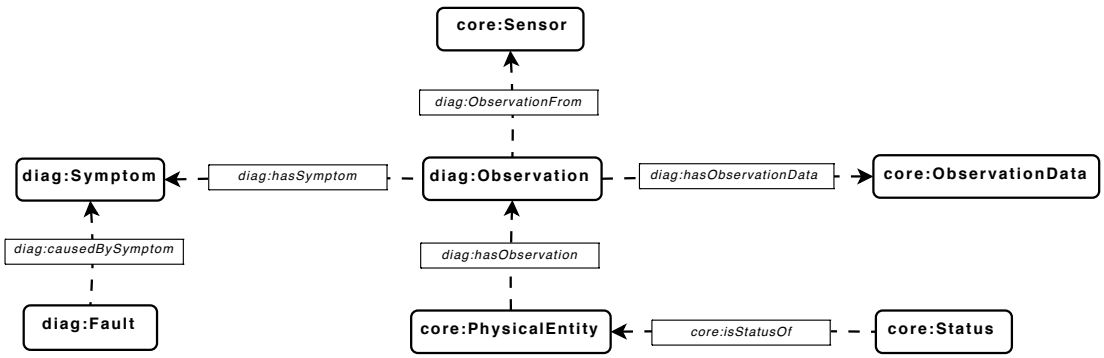


Figure 2.9: Snapshot of the system observation design pattern of the Status Monitoring Application Ontology.

- Event: represents a series of correlated symptoms (low power consumption, position drift), Two subclasses of Events are defined: *CriticalEvent* for high priority events and *IncipientEvent* for the remaining ones.
- Status: links the latest and most updated event information to the systems being monitored (e.g. sidescan transducer),

Note how some of these concepts are related to concepts of the Core Ontology (e.g. an *observation* comes from a *sensor*). These Core Ontology elements provide the exchange of knowledge between service-oriented agents. This is shown during the illustration of operation described in Section 2.5.

2.4.2 Planning Application Ontology

The Plan Application Ontology is used to express the SA_V of the mission planner agent. It is based on concepts and axioms defined in the *Planning Domain Definition Language* (PDDL). The PDDL language was originally created by Ghallab et al. (1998) to standardise the modelling of domains for planning. Concepts are extracted from the language vocabulary and the language grammar is used for describing the relationships and constraints between these concepts. The language takes the same role as the input to STRIPS (Fikes and Nilsson, 1971) and HTN (Sacerdoti, 1975) planning systems, but is vastly more expanded and more flexible. For instance, in its latest version it contains extensions for dealing with extended goals and durative actions. An example of a mission planning scenario described using our variant version of the PDDL language can be found in Appendix B. Our version supports some of the features of PDDL 2.1, such as typing, fluents and discrete durative actions. Additionally, it extends this description

in order to be able to cover a domain-dependent static and dynamic representation of the mission environment.

This ontology represents the domain model Σ of the mission environment Π introduced in Section 1.4. In this section we formally describe this model as its elements will become widely used in the oncoming chapters.

The domain model inside the Planning Application Ontology is defined by the tuple $\Sigma = (C, O_C, V_C, P_V, A_V)$, where:

- $C = \{c_i | i \in \langle 1,2,\ldots,|C|\rangle\}$ is a set of hierarchical classes of objects. All class concepts derive from a root class named `object` (e.g.: `(:types location - object area - location )` $class(c)$ represents the set containing class c and all its ancestors:

$$class(c) = \begin{cases} \{\texttt{object}\}, \textit{ if } c\texttt{=object} \\ \{c\} \cup \{class(parent(c))\}, \textit{ otherwise} \end{cases} \tag{2.1}$$

- $O_C = \{o_j^{c_i} | j \in \langle 1,2,\ldots,|O_C|\rangle, c_i \in C\}$ represents a set of objects or resources of C. Each object o_j is of only one class c_i. Therefore each object o_j belongs to its class and all the ancestors of this class ($\forall o_j^{c_i} \in O_C \;\; class(o_j) = class(c_i)$). (e.g.: `seabedA - area` indicates that the region *seabedA* belongs to the `area`, `location` and `object` classes).

- $V_C = \{v_k^{c_i} | k \in \langle 1,2,\ldots,|V_C|\rangle, c_i \in C\}$ is a set of variables of C. In the same way, each variable v_k belongs to its class and all the ancestors of this class ($\forall v_k^{c_i} \in V_C \;\; class(v_k) = class(c_i)$). (e.g.: `?v - vehicle` indicates a variable v of the class `vehicle`).

  An ordered set of variables and objects defines a list of arguments for a generic element of the domain X:

$$\begin{aligned} arg(\mathrm{X}) = \; & \Big\langle v_k^{c_i} | k \in \langle 1,2,\ldots,n\rangle, \\ & 0 \le n \le |V_C| + |O_C|, c_i \in C \Big\rangle \\ & \subseteq \{V_C \cup O_C\} \end{aligned} \tag{2.2}$$

- $P_V = \{p_m | m \in \langle 1,2,\cdots,|P_V|\rangle\}$ is a set of propositions. A proposition can return a boolean or a numerical value. Each proposition p_m has a list of arguments $arg(p_m)$. (e.g.: `(at ?l - location)` represents the proposition of being at a particular location).

- $F_V = \{f_q, |q \in \langle 1,2,\cdots,|F_V|\rangle\} \subseteq P_V$ is a set of functions. A function is a proposition that returns a numerical value. (e.g.: `(distance ?a ?b - location)` represents the value of the distance between two locations).

- $A_V = \{a_h, |h \in \langle 1,2,\cdots,|A_V|\rangle\}$ is a set of actions. Each action a_h has a list of arguments $arg(a_h)$. An action can have a set of requirements: $condition(a_h) = \{p_m | p_m \in P_V \wedge arg(p_m) \subseteq arg(a_h)\}$, and a set of effects: $effect(a_h) = \{p_m | p_m \in P_V \wedge arg(p_m) \subseteq arg(a_h)\}$. The execution of an action has also a duration in time: $duration(a_h) \in \mathbb{R}$. (e.g.: `(move (?from ?to - location) (:dur (distance ?from ?to))(:cond (at ?from))(:effect (not(at ?from))(at ?to))).`

From this tuple, another two sets can be calculated:

- $R_O = \{r_y^{p_m} | y \in \langle 1,2,\cdots,|R_O|\rangle, p_m \in P_V, arg(r_y^{p_m}) \subseteq O_C\}$ is the set of proposition facts. Proposition facts are propositions of the domain model Σ whose variables have been all grounded to objects in the domain. A proposition fact $r_y^{p_m}$ is an instantiation of a proposition p_m for a particular list of objects as arguments $arg(r_y^{p_m})$. (e.g.: `(at seabedA)`).

- $G_O = \{g_z^{a_h} | z \in \langle 1,2,\cdots,|G_O|\rangle, a_h \in A_V, arg(g_z^{a_h}) \subseteq O_C\}$ is the set of ground actions. Ground actions are actions of the domain model Σ whose variables have been all grounded to objects in the domain. A ground action g_z is an instantiation of an action a_h for a particular list of objects as arguments $arg(r_z^{a_h})$. (e.g.: `(move start seabedA)`).

A mission plan can be seen as a list of ground actions. Figure 2.10 shows how the concept 'Mission' relates to some of the Core Ontology concepts (e.g. a list of *capability* concepts is required to perform a mission *action*).

2.5 Validation of Semantic-based Situation Awareness

We validated this framework during the trials described in Section 6.7. This section illustrates how our semantic-based framework was used during these trials. The scenario for these trials was based on the mine counter measure (MCM) operation scenario using AUVs. In this scenario, AUVs support and provide solutions for mine-hunting and neutralisation. The operation involves high levels of uncertainty and risk of damage to the vehicle.

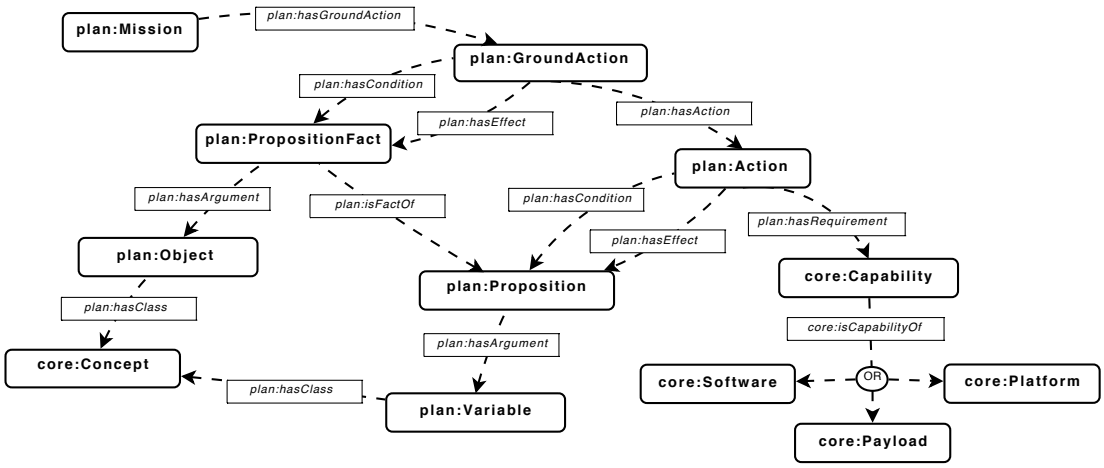


Figure 2.10: Snapshot of the Planning Application Ontology around the concept 'Mission' (top left).

The system was oriented (in the OODA-loop sense) using *a priori* information about the environment and the platform and a declarative description of the goals of the mission. The *a priori* knowledge was represented using the Core Ontology. Knowledge available about the platform configuration capabilities is displayed in Figure 2.11 using Core Ontology concepts. Knowledge about the environment was provided based on automatic computer-aided seabed classification information generated from existing data (Reed et al., 2006b). The declarative description of the mission requirements was represented using concepts from the Planning Application Ontology. This could be summarised as 'survey all known areas maximizing efficiency'. The initial mission plan is displayed in Figure 2.12 using Planning Application Ontology concepts.

2.5.1 Pre-mission Reasoning

Note that the previously described separation between Core knowledge and Planning knowledge gracefully aligns with the separation between platform engineers and mission scientists on current AUV operations. Imagine that the platform capabilities were described in Core Ontology terms by the engineers that manufactured the platform. Using our approach, a scientific operator should be able to describe the mission objectives without understanding the low-level functionalities of the platform. Thus, it is important to assist the operator in knowing if the platform capabilities can match the mission requirements before starting the mission. This can be done by providing an autonomous answer to the following question:

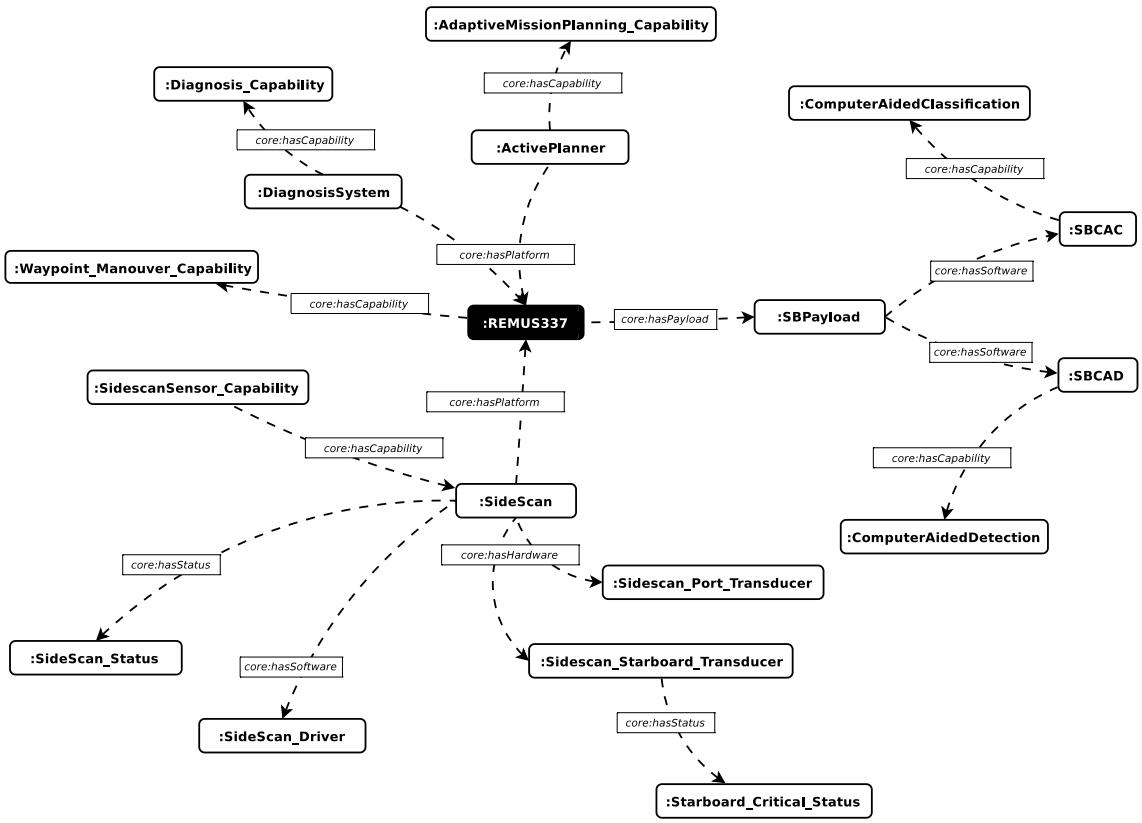


Figure 2.11: Core Ontology instances (boxes) and relations (arrows) for the demonstration scenario. The diagram represents the main platform, its components and their capabilities.

- *Is this platform configuration suitable to successfully perform this mission?*

In order to answer this question, new knowledge could be inferred from the initial Core Ontology orientation. The Core Ontology rule engine was executed, providing this additional knowledge. A set of predefined rules helped to orient the knowledge base and to infer new relationships between instances. An example of a transitive rule dealing with the transfer of payload capabilities to the platform is represented in Eq. 2.3.

$$\begin{aligned} &core:isCapabilityOf(?Capability,?Payload)\wedge \\ &core:isPayloadOf(?Payload,?Platform) \\ &\rightarrow core:isCapabilityOf(?Capability,?Platform) \end{aligned} \tag{2.3}$$

Once all the possible knowledge is extracted, it is possible to query the knowledge base in order to extract the list of capabilities of the platform (see Eq. 2.4) and the list of requirements of the mission (see Eq. 2.5).

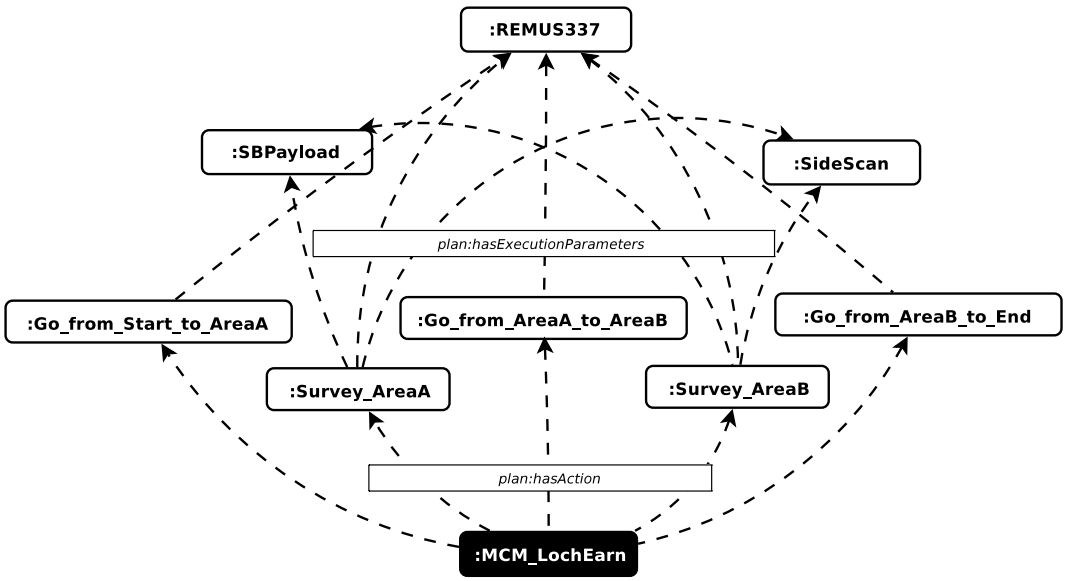


Figure 2.12: Planning Application Ontology concepts representing the mission planning actions, their execution parameters, and their relationships.

$$\begin{aligned} &SELECT \quad ?Platform\ ?Cap \\ &WHERE\ \{ \quad rdf{:}type(\ ?Platform,\ core{:}Platform) \wedge \\ &\qquad\quad core{:}hasCapability(?Platform,?Cap)\ \} \end{aligned} \tag{2.4}$$

$$\begin{aligned} &SELECT \quad ?Mission\ ?Req \\ &WHERE\ \{ \quad plan{:}hasAction(\ ?Mission,\ ?Action) \wedge \\ &\qquad\quad plan{:}hasRequirement(\ ?Action,?Req\ )\ \} \end{aligned} \tag{2.5}$$

This way, it is possible to autonomously extract that the requirements of the mission plan for this experiment are[1]:

- *core:WaypointManeuver_Capability* $\in$ *jaus:Maneuver_Capability*
- *core:ComputerAidedClassification_Capability* $\in$ *jaus:Auto_RSTA-I_Capability*
- *core:ComputerAidedDetection_Capability* $\in$ *jaus:Auto_RSTA-I_Capability*
- *core:SidescanSensor_Capability* $\in$ *jaus:Environmental_Sensing_Capability*

which were a subset of the platform capabilities.

Therefore, for the particular case of this scenario, the platform configuration is suitable to attempt accomplishing the mission objectives.

[1]Auto_RSTA-I, i.e. Autonomous Reconnaissance/Surveillance/Target Acquisition & Identification capability concepts inherited from JAUS (SAE-AS4 AIR5664, 2006)

2.5.2 In-Mission Adaptation

While the mission is being executed, the status monitor agent keeps the knowledge base updated with new observations (in the OODA-loop sense) by reporting changes in the status of hardware components, such as batteries and sensors, and external parameters, such as water currents.

The mission planner adapts the mission to the changes when observations indicate that some of the changes detected are affecting the mission under execution. This need to adapt is detected by the mission planner agent by querying the knowledge base with the following question:

- *Are the observations coming from the environment affecting the mission currently under execution?*

In order to show the reasoning process involved during the event detection, monitoring and response phases of the mission adaptation process, an internal event in the form of a component fault was temporarily simulated in the host vehicle. The fault simulated the gains of the starboard transducer of the sidescan sonar dropping to their minimum levels half way through the performance of the survey action, a lawn mower pattern.

For the detection phase, the low gain signals from the sidescan transducer triggered a symptom instance, which had an associated event level. This event level, represented in the Status Monitoring Application Ontology using a value partition pattern, played a key role in the classification of the instances in the *Event* concept between being Critical or Incipient. This classification is represented axiomatically in Eqs. 2.6 and 2.7.

$$\begin{aligned} \text{status:CriticalEvent} \subseteq\ & \text{status:Event} \wedge \ni\text{status:causedBySymptom} \ldots \\ & (\text{status:Symptom} \wedge \ni\text{status:hasEventLevel} \ldots \\ & (\text{status:Level} \wedge \ni\text{status:High})) \end{aligned} \tag{2.6}$$

$$\begin{aligned} \text{status:IncipientEvent} \subseteq\ & \text{status:Event} \wedge \ni\text{status:causedBySymptom} \ldots \\ & (\text{status:Symptom} \wedge \ni\text{status:hasEventLevel} \ldots \\ & (\text{status:Level} \wedge \ni\text{status:Med})) \end{aligned} \tag{2.7}$$

After the *Event* individuals are re-classified, the *Status* property of the related component in the Core Ontology is updated.

During the diagnosis event phase, a critical status of a component is only considered to be caused by a critical event. Therefore, due to the fact that the sidescan sonar component is composed of two transducers, port and starboard, one malfunctioned transducer is only diagnosed as an Incipient *Status* of the overall sidescan sonar component.

During the response phase, the *Status* property of the Core Ontology components was used by the mission planner to perform the plan diagnosis of the mission under execution. The query to the knowledge base shown in Eq. 2.8 reported that the *survey* action of the two areas in the mission plan are affected by the incipient status of the sidescan sonar.

$$\begin{array}{ll} SELECT & ?Mission\ ?Action\ ?Param\ ?Status \\ WHERE\ \{ & plan{:}hasAction(\ ?Mission,\ ?Action) \wedge \\ & plan{:}hasExecParam(\ ?Action,?Param) \wedge \\ & plan{:}hasStatus(\ ?Param,\ ?Status)\ \} \end{array} \tag{2.8}$$

The adaptive mission planning process performed for dealing with this information is described in Chapter 6.

The same procedure was used after the transducer recovery was reported, to adapt the survey action to the normal pattern during the second lawn mower survey. In a similar process, the system adapted the lawnmower pattern survey of the areas to the detected water current *Status* at the moment of initialising the survey of the areas.

2.6 Summary and Outlook

This chapter presents a semantic-based framework that provides the core architecture for knowledge representation for service oriented agents in autonomous vehicles. The framework combines the initial expert orientation and the observations acquired during mission in order to provide unmanned vehicle situation awareness. This is currently unavailable in autonomous underwater platforms. It has direct impact on the knowledge distribution between embedded agents at all levels of representation.

This work is highly relevant to underwater platforms, especially where autonomy and on-board decision making are required. The approach is extensible to any embedded agent and provides benefits for improving local (service level) and global (system level) situation awareness.

The framework has been applied to the problem of adaptive mission planning. In this scenario, the approach has shown how the status monitor and the mission planner agent can collaborate in order to detect mission plan failures when changes are detected in the mission environment. The approach liberates the operator from the low-level understanding of the platform functionalities. Also, it maintains the agents in their context of expertise and independent from the domain.

Future effort aims to decentralise the current implementation of the approach so agents residing in other platforms can contribute to the knowledge enrichment of the mission environment. These efforts are currently undergoing under the research programs presented in Section 7.4.

2.7 Key Publications

- The requirements for shared situation awareness between agents were identified in the paper *Distributed ontological world model for autonomous multi vehicle operations* presented at the **Conference for Systems Engineering for Autonomous Systems of the Defence Technology Centre** (SEAS-DTC'07) in Edinburgh (Scotland) on July 2007 (Cartwright et al., 2007).

- Results from this chapter were presented in the paper *Semantic knowledge-based representation for improving situation awareness in service oriented agents of autonomous underwater vehicles* during the **International Conference IEEE Oceans** in Quebec (Canada) on September 2008 (Patrón et al., 2008a).

- The framework for semantic situation awareness is described in the article *Semantic knowledge-based framework to improve the situation awareness of autonomous underwater vehicles* accepted for publication at the **Journal of IEEE Transactions on Knowledge and Data Engineering** (Miguelanez et al., 2010)

Chapter 3

Plan Proximity

The distance is nothing; it is only the first step that is difficult.

– Madame Marie du Deffand

3.1 Introduction

In the introduction we presented a list of research objectives. Possibly the most important research objective is being able to measure the adaptive process in order to be able to evaluate the different mission planning strategies. There is currently no accepted way of doing this.

Traditionally, planning is evaluated for completeness and efficiency (Howey et al., 2004). In a dynamically changing mission environment, where a plan adapts along a timeline in response to differences between the expected and observed state of the environment, classical planning metrics do not consider the desired properties of adaptability. Evaluating adaptability requires the comparison of different planning strategies dealing with the adaptation process.

Plan Stability was originally proposed as a measure for comparing two plans generated with different planning strategies that solve the plan adaptation process Fox et al. (2006a). This measure considers the missing actions from the plan used as reference and the extra actions incorporated to the test plan. However, it does not take into account the ordering of these actions, nor does it consider the outcomes of these plans.

This chapter presents a novel metric called Plan Proximity. It compares planning strategies by measuring the difference between the sequence of actions in the plans and the difference between the expected outcome states. We show that this metric provides

a better estimation of the difference between adaptation planning strategies by looking at the proximity between the original plan and the plans that the adaptation strategies generate to replace it. The chapter shows that Plan Proximity, together with the set of all possible plans for a given domain space, makes up a metric space. Also, it presents arguments to support the claim that this metric is more informed than previously accepted metrics for comparing planning strategies solving the adaptation process.

Additionally, the second half of this chapter introduces a set of weights and mechanisms to incorporate semantic information about the domain in the formulation of the metric. This extension is based on the concepts of the domain model defined in the previous chapter as part of the semantic-based unmanned situation awareness.

By making use of two basic assumptions, hierarchy of the classes in the domain and semantic separation of the propositions, the metric remains independent from any domain-specific measurements. Plan Proximity is slightly more strongly correlated with the amount of change introduced than Semantic Plan Proximity. However, the quantity of change is not as meaningful as the types of changes made: Semantic Plan Proximity is more informed than Plan Proximity.

As a result, this chapter provides a solution to the research objective of measuring performance of adaptive mission planning strategies (see Section 1.3.3.5).

3.2 The Dynamic Mission Planning Problem

The situation in which an initial plan has been constructed and the context in which it is being executed has deviated from the original context was termed by Fox et al. (2006a) as the dynamic planning problem. The problem has been addressed by authors, such as Gerevini and Serina (2000a), Horty and Pollack (2001) and Krogt and Weerdt (2005) who have proposed various strategies for managing it. It can be formulated as:

Definition 3.2.1 *Given a tuple* (Π, π, Π')*, where* π *is a mission plan that accomplishes the mission objectives for the mission environment* Π *and* Π' *is a variant of the mission environment for the same domain space, i.e.* $\Pi, \Pi' \in \Theta$*, the dynamic planning problem is the search of a* π' *that accomplishes the mission objectives of* Π'*.*

In which, a mission plan can be defined as:

Definition 3.2.2 *A mission plan* π *is an ordered list of ground actions of length n that is expected to accomplish the mission objectives for a given mission environment* Π

and to transform the state of the domain from an initial state I to a final state G.

and a state of the mission environment is defined as:

Definition 3.2.3 *A state s is a set of proposition facts describing a configuration of the mission environment* Π.

3.3 Comparison of Planning Strategies

Firstly, this section identifies Plan Stability to be the main related work for comparing planning strategies solving the dynamic planning problem. The section highlights the flaws encountered in Plan Stability.

In the second part, this section introduces a novel metric called Plan Proximity. This new metric overcomes the flaws encountered in Plan Stability. As a consequence, Plan Proximity is better informed in measuring the adaptation process between plans.

3.3.1 Plan Stability

A reference plan π_1 and a test plan π_2 can be compared by looking at the different number of ground actions that they contain. In this way, Fox et al. (2006a) defined the Plan Stability related to the measure of the difference between two plans as:

Definition 3.3.1 *Given a reference plan,* π_1 *and a test plan,* π_2*, the difference between* π_1 *and* π_2*,* $D(\pi_1,\pi_2)$ *is the number of ground actions that appear in* π_1 *and not in* π_2 *plus the number of ground actions that appear in* π_2 *and not in* π_1.

This difference does not take into account the point at which these ground actions are performed. However, this is important during plan execution. Because, even if all the mission objectives are achieved, this can drastically change the outcome of the plan. Thus, ordering is important when comparing two plans.

Ref	$Test_1$
A	B
B	A

Table 3.1: Example of a reference plan (A B) and a test plan (B A) highlighting the need of a metric that captures the order of ground actions when measuring the adaptation process.

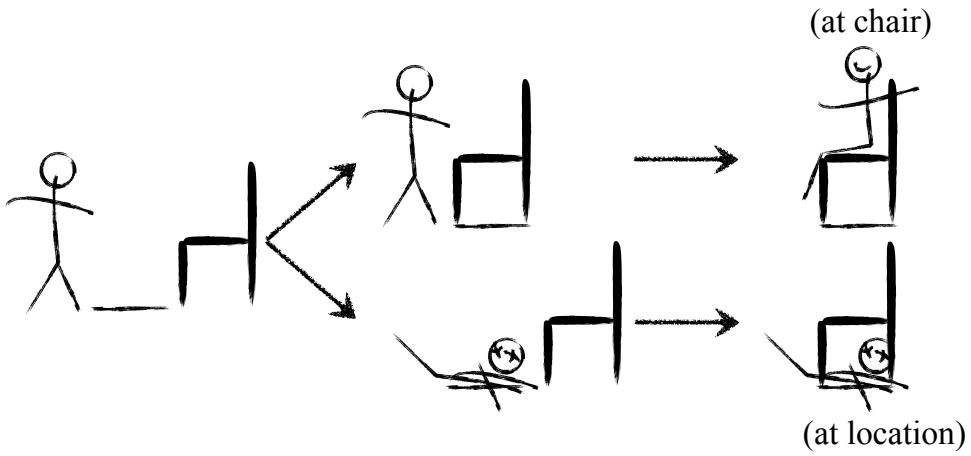


Figure 3.1: A metric comparing plans needs to consider the ordering of ground actions in the plans. In an scenario starting with a person, a chair and a location, it can be seen that the final state differs considerably based on the ordering of execution: (A B) (top) or (B A) (bottom) sequence of execution for reference and test plan of Table. 3.1 using the ground actions from Eq. 3.1.

For the example shown in Table 3.1, there are no actions that appear in the reference plan and not in the test plan or vice versa. As a result, Plan Stability = 0. This output incorrectly indicates that both plans are equal. However, if actions A and B are the actions described in Eq. 3.1 for the example scenario illustrated in Figure 3.1, it can be seen that the outcomes of the two plans are very different.

$$\begin{aligned} A &= \texttt{(toPlace chair left\_hand location)} \\ B &= \texttt{(toSit slowly)} \end{aligned} \tag{3.1}$$

Note from this example that these metrics comparing plans do not look at the completeness or efficiency of the final plan as studied in Howey et al. (2004). This example aims only to emphasise the impact that swapping the order of the actions in a plan might have in the final outcome, independently on the validity of the plan.

On the other hand, the opposite situation may also occur: completely different plans may leave the environment in a similar state when executed. Therefore, it is also important to take the difference between final states into account when measuring the adaptation process.

In the example shown in Table 3.2, all the ground actions that appear in the reference plan do not appear in the test plan and vice versa. As a result, Plan Stability = 4.

Ref	$Test_2$
A	C
B	D

Table 3.2: Example of a reference plan (A B) and a test plan (C D) highlighting the need of a metric that captures the differences between the final outcome states.

This output incorrectly indicates that the two plans are completely different. However, if ground actions A, B are the ones described in Eq. 3.1 and C and D are the ground actions described in Eq. 3.2 for the same example scenario, it can be seen that the outcomes of the two plans are the same (see Figure 3.2).

$$\begin{aligned} C &= \texttt{(toPlace chair right\_hand location)} \\ D &= \texttt{(toSit fast)} \end{aligned} \tag{3.2}$$

Finally, Plan Stability values are dependent on the length of the plans compared. Because these values are not normalised, comparing plans from different mission environments of the same domain space becomes problematic.

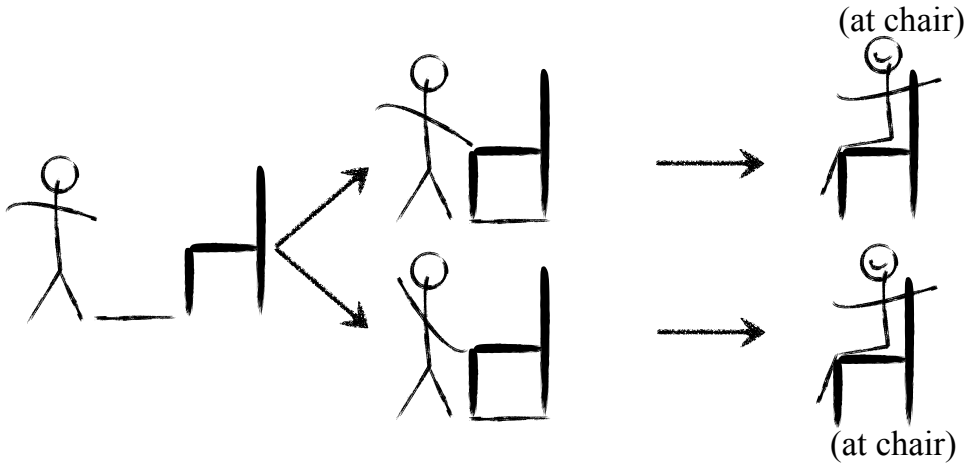


Figure 3.2: A metric comparing plans needs to consider the final outcome state produced by the plans. In an scenario starting with an person, a chair and a location, it can be seen that the final state achieved can be the same when executing a different set of ground actions: (A B) (top) and (C D) (bottom) sequence of execution for reference and test plan of Table. 3.1 using the ground actions from Eq. 3.1 and Eq. 3.2 respectively.

3.3.2 Plan Proximity

This section proposes Plan Proximity as a novel metric that overcomes the flaws encountered in Plan Stability. This metric captures the ordering of the ground actions in the plan and the differences between the outcome states. The values provided by this metric are normalised, and together with the set of all possible plans for a given domain space Θ, make up a metric space. Together, all these factors provide a metric capable of better summarising the difference between two plans. This metric is formed by the plan difference and the outcome state difference.

3.3.2.1 Plan Difference

Several metrics exist in the literature capable of capturing the differences of two ordered sequences of elements (Damerau, 1964; Levenshtein, 1966; Navarro, 2001). Generally, they look at the number of insertions, deletions and permutations required to transform one sequence into the other.

Following another approach, the 'diff' algorithm (Hunt and McIlroy, 1976) solves the longest common subsequence problem. It is commonly used to find the lines that do not change between two files.

This second approach can be applied to capture blocks of ground actions that do not align with the longest common subsequence between the two plans. Thus, it can compare two plans by finding the identically ordered items that are present between two plans.

Definition 3.3.2 *Given a reference plan π_1 and a test plan π_2, the plan difference between π_1 and π_2, $D_p(\pi_1,\pi_2)$, is the number of missing ground actions m_p from the reference plan π_1 and the number of extra ground actions e_p from a test plan π_2 that do not appear in the longest common subsequence of actions.*

$$D_p(\pi_1,\pi_2) = m_p + e_p \tag{3.3}$$

The plan difference is normalised using the sum of the number of actions of the reference plan n_1 and the test plan n_2.

$$\hat{D}_p(\pi_1,\pi_2) = \frac{D_p(\pi_1,\pi_2)}{n_1+n_2} \in [0;1] \tag{3.4}$$

3.3.2.2 State Difference

The ordering of the proposition facts in a state does not change the state. If M is the total number of proposition facts in a domain, a state can be represented as a binary string of length M containing 1's on the position of the string that corresponds to the proposition facts that its configuration represents. The number of proposition facts for a state is represented by m ($m \leq M$). From this representation, the difference between two states can be calculated using the Hamming distance (Hamming, 1950). The Hamming distance between two strings of equal length is the number of positions for which the corresponding symbols are different.

Definition 3.3.3 *Given a reference state s_1 and a test state s_2 for a common domain, the state difference, $D_s(s_1, s_2)$, can be calculated as the Hamming distance between the string representation of s_1 and s_2.*

$$D_s(s_1, s_2) = \sum_{i=1}^{M} x_i \text{ where } x_i = \begin{cases} 0 \text{ if } s_1(i) = s_2(i) \\ 1 \text{ otherwise} \end{cases} \tag{3.5}$$

The state difference is normalised using the sum of the string length of elements in the reference state m_1 and the test state m_2.

$$\hat{D}_s(s_1, s_2) = \frac{D_s(s_1, s_2)}{m_1 + m_2} \in [0; 1] \tag{3.6}$$

3.3.2.3 Plan Proximity

The proximity between a reference plan and a test plan is the combination of the plan difference and the difference of the estimated final states.

Definition 3.3.4 *Given a reference plan π_1 and a test plan π_2, their plan proximity, $PP(\pi_1, \pi_2)$, is the normalised balanced sum of the plan difference $D_p(\pi_1, \pi_2)$ and the state difference of the estimated final states that they are expected to produce $D_s(G_1, G_2)$.*

$$\begin{aligned} PP_\alpha(\pi_1, \pi_2) = \quad & 1 \quad -\alpha \cdot \hat{D}_p(\pi_1, \pi_2) \\ & \quad -(1-\alpha) \cdot \hat{D}_s(G_1, G_2) \in [0; 1] \end{aligned} \tag{3.7}$$

where $\alpha \in [0; 1]$ represents a *balance factor* between plan and state difference. The logical default value of α would be 0.5. However, it will depend on the particular evaluation of plans for the different domains. For instance, depending on the domain, one

might want to give more importance to the plan difference than to the final state difference or vice versa. Plan proximity can be interpreted as the percentage of similarity between two plans. Because plan and state differences are based on distance-based metrics, Plan Proximity also satisfies the distance function properties. This is demonstrated in Appendix A. In consequence, together with the set of all possible plans for given domain space Θ makes up a metric space.

The final states for the reference and test plans presented in Table 3.1 and Table 3.2 are shown in Eq. 3.8.

$$\begin{aligned} G_{Ref} &= \texttt{(at chair location) (at chair)} \\ G_{Test_1} &= \texttt{(at chair location) (at location)} \\ G_{Test_2} &= \texttt{(at chair location) (at chair)} \end{aligned} \tag{3.8}$$

The Plan Proximity value for the scenario presented in Table 3.1 is presented in Eq. 3.9.

$$PP_{0.5}(Ref, Test_1) = 1 - 0.5 \times \frac{2}{4} - (1 - 0.5) \times \frac{2}{4} = 0.50 \tag{3.9}$$

Similarly, the Plan Proximity value for the scenario presented in Table 3.2 is presented in Eq. 3.10.

$$PP_{0.5}(Ref, Test_2) = 1 - 0.5 \times \frac{4}{4} - (1 - 0.5) \times \frac{0}{4} = 0.50 \tag{3.10}$$

In comparing these results with the ones provided by Plan Stability, it can be seen that Plan Proximity better captures the difference between the reference plan and the two test plans.

Note that Plan Proximity considers the proposition facts in the states but do not considers values of the functions in those states. This limitation also applies to Plan Stability. As a consequence, numerical values, such as different resource usage between plans, can not be compared by using these metrics.

3.3.2.4 Dynamic Planning Strategy Comparison

The Plan Proximity metric space can be used to compare different planning strategies aiming to solve the dynamic planning problem. Plans coming out of these strategies can be compared and verified against a plan used as reference.

Definition 3.3.5 *Given a dynamic planning problem* (Π, π_0, Π')*, a planning strategy* P_1*, achieves greater plan proximity than another planning strategy* P_2*, if the plans produced by* P_1 *and* P_2*,* π_1 *and* π_2 *respectively, satisfy* $PP(\pi_0, \pi_1) > PP(\pi_0, \pi_2)$.

Based on this definition, the results from Eq. 3.9 and Eq. 3.10 indicate that a plan strategy generating the test plan $Test_1$ of Table 3.1 from the reference plan achieves the same plan proximity as the planning strategy used to generate the test plan $Test_2$ of Table 3.2.

The simplicity of having a unique final value for comparison of plans and plan strategies is generally beneficial. However, situations like the previous example highlight the consequences of obtaining this single value. By balancing Plan Difference and State Difference to obtain Plan Proximity, relevant information gets diluted. Under these situations, the selection of the adequate balance factor for a given domain can become critical. Furthermore, in some situations maintaining the plan and state differences separated may be necessary.

The description of this metric as a tool for comparing different planning strategies solving the dynamic mission planning problem was presented in Patrón and Birch (2009).

3.4 Evaluation of Plan Proximity

A random mission planning problem was created to compare the performance of Plan Stability and Plan Proximity when measuring the plan adaptation process. This environment contained a data set with 100 ground actions and 100 propositions facts.

For simplicity, it was assumed that all ground actions could be executed at any time, i.e. the actions do not have preconditions and can be executed at any state. This simplification was made possible by the fact that we focussed in measuring the difference between plans and not at their validity. As a consequence, we did not required consistency between the preconditions and effects of the ground actions manipulated during the insertions, permutations and deletions. This considerably simplified the way the plans could be randomly generated for the experiment without invalidating the purpose of the experiment. Each ground action has a set of positive effects and a set of negative effects over the domain. Each of these sets is selected randomly from the proposition fact set. The maximum number of positive and negative effects that a ground action can contain is also limited to 100.

A reference plan π_1 is generated as a sequence of ground actions randomly selected from the action set. The reference plan length n_1 is randomly chosen and limited to a maximum of 100 ground actions. The reference final state s_1 is calculated by executing the ground actions from π_1 in sequence from an initial empty state. At each step of this calculation, the positive and negative effects of the ground action are applied to the state of the previous step.

A test plan π_2 is calculated by applying a random number of changes κ to the reference plan π_1. The maximum number of changes applied to calculate π_2 from π_1 is limited to 100. These changes are randomly selected from the following three types:

- **Insertion** of a new ground action randomly selected from the action set.
- **Deletion** of a ground action from a random position from the current adapted plan.
- **Permutation** of two random ground actions from the current adapted plan.

Thus, the adaptation changes are divided between the number of ground actions inserted (μ), the number of ground actions deleted (ν), and the number of permutations (ω).

$$\kappa = \mu + \nu + \omega \tag{3.11}$$

Once π_2 has been computed, the test final state s_2 is calculated, similarly as for s_1; The ground actions of the test plan π_2 are executed in sequence from an empty initial state.

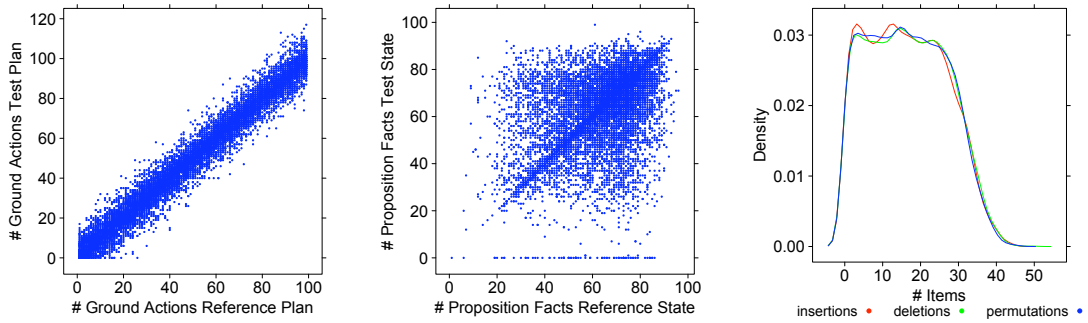
Finally, Plan Stability and Plan Proximity are calculated over the plan-state pairs $[\pi_1, s_1]$ and $[\pi_2, s_2]$.

The proportion of number of changes introduced κ versus the total number of ground actions handled in the process is also calculated. The total number of ground actions is the ground actions originally in π_1, n_1, and the ground actions inserted during the adaptation process, μ. This proportion is called the Change-Action ratio:

$$\textit{Change-Action ratio} = \frac{\kappa}{n_1 + \mu} \tag{3.12}$$

The metrics that we are calculated are designed to measure the difference between plans by looking only at the plans and at their final states. On the other hand, the Change-Action ratio is a metric that has explicit access to the way the plans have been

calculated and how the changes were introduced. Therefore, the Change-Action ratio is a suitable reference to be used as ground truth during the evaluation process. This process was repeated 10000 times.



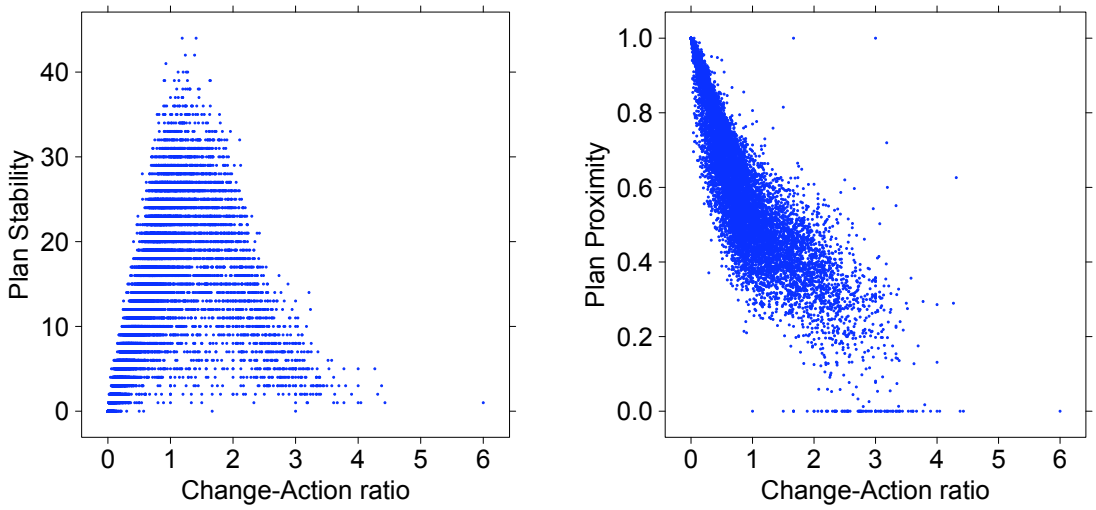
a) b) c)

Figure 3.3: a) Variability of number of actions in the test plans (n_2) in relation to the actions in the reference plans (n_1). b) Variability of number of proposition facts in the final state of the test plan (s_2) in relation to the number of proposition facts in the final state of the reference plan (s_1). c) Density distribution of adaptation changes (κ) is evenly distributed between insertions (μ), deletions (ν) and permutations (ω) of actions over the 10000 sample data set.

Figure 3.3.a shows the variability of plan length between the reference plan length n_1 and the test plan length n_2. The adaptation process introduces even more variability in the amount of proposition facts present at the final states (see Figure 3.3.b). Figure 3.3.c shows the density distribution of the changes introduced. It can be seen how the random selection of changes introduced during the adaptation experiment is evenly distributed between insertions, deletions and permutations of ground actions.

Figure 3.4 shows the distribution of Plan Stability and Plan Proximity over the adaptation Change-Action ratio. The significance of the two measurements to explain the adaptation process can be extracted by analysing the correlation between the variables.

Pearson's correlation coefficient (Pearson, 1920) is the most common measurement of statistical dependency. However, it is sensitive to a linear relationship between two variables. Rank correlation coefficients offer an alternative when this linearity can not be assumed. They assess a relationship described using a monotonic function, that does not have to be necessarily linear. Spearman's rank correlation coefficient (Spearman, 1904) and Kendall's rank correlation coefficient (Kendall, 1938) are rank correlation



a) b)

Figure 3.4: a) Distribution of Plan Stability over the adaptation Change-Action ratio. b) Distribution of Plan Proximity metric over the adaptation Change-Action ratio.

coefficients. For this evaluation, we have used the Spearman's ρ rank correlation coefficient.

	Plan Stability	Plan Proximity
Change-Action ratio	0.609	-0.889

Table 3.3: Spearman's ρ rank correlation coefficient for Plan Stability and Plan Proximity in relation to the adaptation Change-Action ratio. It can be seen that Plan Proximity correlation with the Change-Action ratio exceeds that of Plan Stability, i.e. absolute value of ρ closer to 1.

Table 3.3 shows the Spearman's ρ rank correlation coefficient values of the relationship between the measures and the adaptation ratio. In these results, the correlation of Plan Proximity with the Change-Action ratio is maintained even when high levels of adaptation are introduced, i.e. Change-Action ratio $\gg 1$. These results induce to conclude that Plan Proximity metric presents a much better explanation of the adaptation process than Plan Stability. Therefore, it is a preferable metric for comparing planning strategies solving the dynamic planning problem. As a consequence, in the following chapters this metric is used between valid plans in order to compare the different planning strategies studied.

3.5 Semantic Comparison of Planning Strategies

3.5.1 Syntactic Plan Proximity

The previous section showed how correlated the Plan Proximity metric is to the ratio of adaptability introduced in a plan. However, this metric is still only a syntactic approach to the comparison of plans. This approach still lacks any real ability to characterize plan difference in a common-sense way, primarily because it is trying to be entirely domain-independent; most real problems with plan distance elements are distinctly not uniform in their concern for different actions or state features/propositions.

For example, if some extra grab-images actions are inserted to a mission plan, it is fair to say that the new plan is more similar to the old one than if the same number of extra grab-objects actions are added, as one can consider that with the latter set of actions the mission environment is more disturbed. While the state differencing part of Plan Proximity metric captures that, it will not notice the difference between substituting a karabiner for a shackle, vs. substituting it for some sea weed. These are domain-specific differences requiring rich semantic knowledge to understand.

3.5.2 Semantic Plan Proximity

The focus of this section is to incorporate semantic knowledge in the metric formulation while still remaining domain independent. Using the planning concepts presented in Section 2.4.2, this section incorporates this knowledge into the formulation by considering the inclusion of additional weights and mechanisms.

In order to do this, we have made two assumptions about the domain:

1. There is a tree hierarchy of classes in the domain model structuring the objects in the domain. This is captured by the domain model in Eq. 2.1.

2. The definition of propositions in the domain model is expected to be as semantically separated as possible, i.e. there are not different propositions containing similar semantic meaning in the domain model.

These two assumptions allow the metric to be independent from the domain, while at the same time capturing some of its semantics. The approach for planning knowledge representation introduced in Section 2.4.2 follows these assumptions.

3.5.2.1 Class Distance

We assumed that classes of objects are taxonomically organized in a hierarchy in the domain model (see Section 2.4.2). This hierarchy defines a class tree from which it is possible to define a distance measure between classes.

Definition 3.5.1 *The depth of a class is the length of the path to the root class or the hierarchy tree. The level between two classes is defined as the absolute difference between their depths in the hierarchy tree.*

$$level(c_1, c_2) = |depth(c_1) - depth(c_2)| \tag{3.13}$$

Definition 3.5.2 *Given a reference class c_1 and a test class c_2 on a hierarchical class structure of a common domain, the class distance $D_c(c_1, c_2)$ can be calculated as the sum of the levels from c_1 and c_2 to the first common ancestor class p.*

$$D_c(c_1, c_2) = level(c_1, p) + level(c_2, p) \tag{3.14}$$

This distance can be normalised by dividing it by twice the maximum depth l of the hierarchy tree.

$$\hat{D}_c(c_1, c_2) = \frac{D_c(c_1, c_2)}{2 \times l} \in [0; 1] \tag{3.15}$$

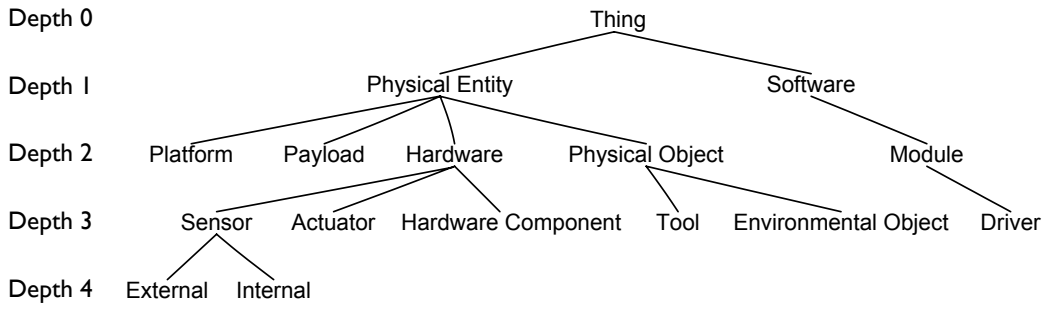


Figure 3.5: Example of a hierarchical tree of classes of a common subset of objects typically found in the domain of unmanned vehicles.

Figure 3.5 shows an example of a hierarchical tree for the classes of objects extracted from a Core Ontology related to the domain of unmanned vehicles. For example, following the previous definition, it can be seen that the normalised class distance between `Sensor` class and `Tool` class is smaller than between `Sensor` class and `Driver` class. Even if they all belong to the same level in the tree (see Eq. 3.16).

$$\hat{D}_c(\texttt{Sensor},\texttt{Tool}) = \frac{2+2}{2\times 4} = \frac{4}{8} = 0.50$$
$$\hat{D}_c(\texttt{Sensor},\texttt{Driver}) = \frac{3+3}{2\times 4} = \frac{6}{8} = 0.75 \quad (3.16)$$

3.5.2.2 Object Distance

The class distance D_c serves as a basis for defining the object distance D_o between two objects of the same domain.

Definition 3.5.3 *Given a reference object o_1 and a test object o_2 for a common domain, the normalised object distance $\hat{D}_o(o_1,o_2)$ is the balanced sum of the object difference $d_o(o_1,o_2)$ and the class distance between the o_1 primary class ($class(o_1)$) and o_2 primary class ($class(o_2)$).*

$$\hat{D}_o(o_1,o_2)_{\alpha_o} = \alpha_o \times d_o(o_1,o_2) + (1-\alpha_o) \times \hat{D}_c(class(o_2),class(o_2)) \in [0;1] \quad (3.17)$$

where $\alpha_o \in [0;1]$ represents the *object balance factor* between the object difference and class distance. In the experiments of Section 3.6, a value of $\alpha_o = 0.25$ has been used in order to compensate with the stronger contribution that the binary object difference result can bring to the equation. This provides small distances for objects that are different but belong to the same class.

The object difference $d_o(o_1,o_2)$ is 0 only if the two objects are equal (see Eq. 3.18).

$$d_o(o_1,o_2) = \begin{cases} 0 \text{ if } o_1 = o_2 \\ 1 \text{ otherwise} \end{cases} \quad (3.18)$$

For example, given the objects $\texttt{sidescan},\texttt{camera} \in \texttt{Sensor}$ and $\texttt{dvl\_driver} \in \texttt{Driver}$ from the domain represented in Figure 3.5, the distances between them can be calculated as in Eq. 3.19.

$$\hat{D}_o(\texttt{sidescan},\texttt{sidescan})_{0.25} = 0.25\times 0 + (1-0.25)\times(\frac{0+0}{2\times 4}) = 0.00$$
$$\hat{D}_o(\texttt{sidescan},\texttt{camera})_{0.25} = 0.25\times 1 + (1-0.25)\times(\frac{0+0}{2\times 4}) = 0.25$$
$$\hat{D}_o(\texttt{sidescan},\texttt{dvl\_driver})_{0.25} = 0.25\times 1 + (1-0.25)\times(\frac{3+3}{2\times 4}) = 0.81 \quad (3.19)$$

3.5.2.3 Attribute List Distance

The object distance D_o serves as a basis for defining the attribute list difference D_v between two lists of attributes containing objects from the same domain.

Definition 3.5.4 *Given a reference list of attributes v_1 of length m_1 and a test list of attributes v_2 of length m_2 for a common domain, the attribute list difference can be calculated as the sum of object distances between the attributes located at the same positions in the lists.*

$$D_v(v_1, v_2) = \sum_{i=1}^{\min(m_1,m_2)} \hat{D}_o(v_1^i, v_2^i) + \sum_{i=\min(m_1,m_2)+1}^{\max(m_1,m_2)} 1 \tag{3.20}$$

This difference can be normalised by dividing it by the maximum length of the two lists.

$$\hat{D}_v(v_1, v_2) = \frac{D_v(v_1, v_2)}{max(m_1, m_2)} \in [0; 1] \tag{3.21}$$

3.5.2.4 Fact Distance

The second assumption in Section 3.5.2 stated that propositions of the domain are defined in the domain model by being semantically exclusive. From this, it is possible to define a distance measure between facts:

Definition 3.5.5 *Given a reference fact r_1 and a test fact r_2 for a common domain, the fact distance $D_r(r_1, r_2)$ can be calculated as the balanced sum of the proposition difference $d_p(proposition(r_1), proposition(r_2))$ and the attribute list difference of the fact arguments $D_v(arg(r_1), arg(r_2))$.*

$$\begin{aligned} D_r(r_1, r_2) = \quad & \alpha_r \times d_p(proposition(r_1), proposition(p_2)) \\ & + (1 - \alpha_f) \times D_v(arg(r_1), arg(r_2)) \in [0; 1] \end{aligned} \tag{3.22}$$

where $\alpha_r \in [0; 1]$ represents the *fact balance factor* between the proposition difference and the attribute list distance. In the experiments of Section 3.6, a value of $\alpha_r = 0.25$ has been used. As discussed in Section 3.5.2.2, this compensates with the stronger contribution that the binary proposition difference result can bring to the equation.

The proposition difference $d_p(p_1, p_2)$ is 0 only if the two resources are equal (see Eq. 3.23).

$$d_p(p_1, p_2) = \begin{cases} 0 \text{ if } p_1 = p_2 \\ 1 \text{ otherwise} \end{cases} \tag{3.23}$$

As an example, Table 3.4 shows the fact distances $\hat{D}_r$ between the reference fact r_1 and different test facts presented in Eq. 3.24.

$$\begin{aligned} r_1 &= \texttt{(installed sidescan vehicleA)} \\ r_{2a} &= \texttt{(installed sidescan vehicleB)} \\ r_{2b} &= \texttt{(installed camera vehicleA)} \\ r_{2c} &= \texttt{(active sidescan vehicleA low\_frequency)} \\ r_{2d} &= \texttt{(captured rock gripper)} \end{aligned} \tag{3.24}$$

$\hat{D}_r$	r_{2a}	r_{2b}	r_{2c}	r_{2d}
r_1	0.09	0.19	0.50	0.58

Table 3.4: Fact distance $\hat{D}_r$ between a reference fact example and different test facts.

3.5.2.5 Ground Action Distance

In a similar way, the distance between two ground actions can be calculated.

Definition 3.5.6 *Given a reference ground action g_1 and a test ground action g_2 for a common domain, the ground action distance $D_g(g_1,g_2)$ is the balanced sum of the action difference $d_a(action(g_1),action(g_2))$ and the attribute list difference of the ground action arguments $D_v(arg(g_1),arg(g_2))$.*

$$\begin{aligned} D_g(g_1,g_2) = \; & \alpha_g \times d_a(action(g_1),action(g_2)) \\ & +(1-\alpha_g) \times \hat{D}_v(arg(g_1),arg(g_2)) \end{aligned} \tag{3.25}$$

where $\alpha_g \in [0;1]$ represents the *ground action balance factor* between the action difference and the attribute list distance. In the experiments of Section 3.6, a value of $\alpha_g = 0.25$ has been used. As discussed in previous sections, this compensates with the stronger contribution that the binary action difference result can bring to the equation.

The action difference $d_a(a_1,a_2)$ is 0 only if the two actions are equal (see Eq. 3.26).

$$d_a(a_1,a_2) = \begin{cases} 0 \text{ if } a_1 = a_2 \\ 1 \text{ otherwise} \end{cases} \tag{3.26}$$

Note that, unlike for propositions, we have not made any assumption about the semantic overlap of actions. As we discussed in Section 3.5.1, the semantic difference

of ground actions, such as the ones shown in the example of Eq. 3.27, is captured by the state differencing part of the metric by looking at the outcome state that they generate.

$$\begin{aligned} g_1 &= \texttt{(toTransfer vehicle1 locationA locationB)} \\ g_2 &= \texttt{(toMove vehicle1 locationA locationB)} \end{aligned} \tag{3.27}$$

Also note that this distance has not been used in the Semantic Plan Proximity metric described in Section 3.5.2.8. This distance is only used in Section 3.6 as a way to evaluate the new metric in measuring the adaptation process.

3.5.2.6 Semantic State Difference

The semantic state difference measures the semantic difference between two states by looking at the fact distances of their elements.

Definition 3.5.7 *Given a reference state s_1 and a test state s_2 for a common domain, the semantic state difference $SD_s(s_1,s_2)$ can be calculated as the sum of the minimum distances between the proposition facts in s_1 and the proposition facts in s_2.*

This can be calculated recursively as:

$$SD_s(s_1,s_2) = \begin{cases} \min\limits_{\substack{0<i\leq m_1 \\ 0<j\leq m_2}} (D_r(r_1^i, r_2^j)) & \text{if } m_1 = 1 \text{ or } m_2 = 1 \\ \min\limits_{\substack{0<i\leq m_1 \\ 0<j\leq m_2}} (D_r(r_1^i, r_2^j) + SD_s(s_1 \setminus r_1^i, s_2 \setminus r_2^j)) & \text{otherwise} \end{cases} \tag{3.28}$$

The semantic state difference is normalised using the length m_1 of the state s_1 and the length m_2 of the state s_2.

$$\hat{SD}_s(s_1,s_2) = \frac{SD_s(s_1,s_2) + \sum\limits_{\min(m_1,m_2)}^{\max(m_1,m_2)} 1}{m_1 + m_2} \in [0;1] \tag{3.29}$$

3.5.2.7 Semantic Plan Difference

The semantic plan difference measures the semantic difference between two plans by looking at the state differences of the changes detected between the two plans.

Definition 3.5.8 *Given a reference plan π_1 and a test plan π_2, the semantic plan difference between π_1 and π_2, $SD_p(\pi_1,\pi_2)$ is the total sum of the semantic state differences*

produced by the number of missing actions m_p from the reference plan π_1 and the number of extra actions e_p from a test plan π_2 produced by each of the change subsequences that do not appear in the longest common subsequence of actions.

$$SD_p(\pi_1,\pi_2) = \sum_{i=0}^{changes(diff)} (m_p^i + e_p^i) \times \hat{SD}_s(s_{m_p^i}, s_{e_p^i}) \tag{3.30}$$

where $changes(diff)$ is the set of subsequences produced by the 'diff' algorithm around the longest common subsequence of actions that are not part of this longest subsequence. For each of these change subsequences i, m_p^i corresponds to the missing actions from the reference plan and e_p^i represents the extra actions from the test plan. The substates produced by these two subsequences of actions are $s_{m_p^i}$ and $s_{e_p^i}$ respectively.

The semantic plan difference is normalised using the sum of the number of actions of the reference plan n_1 and the test plan n_2.

$$\hat{SD}_p(\pi_1,\pi_2) = \frac{SD_p(\pi_1,\pi_2)}{n_1+n_2} \in [0;1] \tag{3.31}$$

3.5.2.8 Semantic Plan Proximity

The semantic plan proximity between a reference plan and a test plan is the combination of the semantic plan difference of the plans and the semantic state difference of the estimated final states.

Definition 3.5.9 *Given a reference plan π_1 and a test plan π_2, their semantic plan proximity $SPP(\pi_1,\pi_2)$ is the normalised balanced sum of the semantic plan difference $SD_p(\pi_1,\pi_2)$ and the semantic state difference of the estimated final states that they are expected to produce $SD_s(G_1,G_2)$.*

$$\begin{aligned} SPP_\alpha(\pi_1,\pi_2) = \quad & 1 \quad -\alpha \cdot \hat{SD}_p(\pi_1,\pi_2) \\ & \quad -(1-\alpha) \cdot \hat{SD}_s(G_1,G_2) \in [0;1] \end{aligned} \tag{3.32}$$

where $\alpha \in [0;1]$ represents a *balance factor* between the semantic plan difference and the semantic state difference. In the experiments of Section 3.6, a value of $\alpha = 0.5$ have been used. However, as stated in Section 3.3.2.4, the selection of this factor will depend on the particular evaluation of plans for the different domains. Semantic Plan Proximity can be interpreted as the percentage of semantic similarity between two plans.

3.5.2.9 Dynamic Planning Strategy Comparison

Semantic Plan Proximity can be used to compare different planning strategies aiming to solve the dynamic planning problem. Plans coming out of these strategies can be compared and verified against a reference plan.

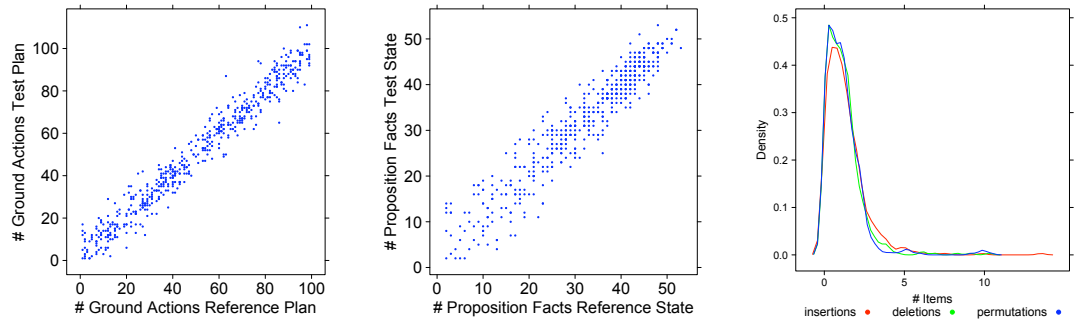
Definition 3.5.10 *Given a dynamic planning problem* (Π, π_0, Π')*, a planning strategy,* P_1*, achieves greater semantic plan proximity than a planning strategy,* P_2*, if the plans produced by* P_1 *and* P_2*,* π_1 *and* π_2 *respectively, satisfy* $SPP(\pi_0, \pi_1) > SPP(\pi_0, \pi_2)$.

3.6 Evaluation of Semantic Plan Proximity

A random mission planning problem was created to compare the performance of Semantic Plan Proximity and Plan Proximity when measuring the plan adaptation process. This environment contained a data set with 32 objects distributed among 10 classes in a hierarchy tree, 50 propositions and 20 actions. The number of arguments of the resources was randomly selected from the set of classes. This generated a total of 320 proposition facts. Each action had a set of positive effects and a set of negative effects. Each of these sets was selected randomly from the proposition fact set. The maximum number of positive and negative effects that an action could contain was limited to 6. This generated a total of 4140 ground actions. For simplicity, it was assumed that all ground actions could be executed at any time, i.e. the actions do not have preconditions and can be executed at any state.

A reference plan π_1 is generated as a sequence of ground actions randomly selected from the ground action set. The reference plan length n_1 is randomly chosen and limited to a maximum of 100 ground actions. The reference final state s_1 is calculated by executing the ground actions from π_1 in sequence from an empty initial state. At each step of this calculation, the positive and negative effects are applied to the previously calculated state.

A test plan π_2 is calculated by applying a random number of changes κ to the reference plan. The maximum number of changes applied to calculate π_2 from π_1 is limited to 100. These changes are randomly selected from the three types – insertion, deletion, and permutation – described in Section 3.4. Once π_2 has been computed, the final test state s_2 is calculated, similarly as for s_1: the ground actions of the test plan π_2 are executed in sequence from an initial empty state.



a) b) c)

Figure 3.6: a) Variability of number of ground actions in the test plans (n_2) in relation to the ground actions in the reference plans (n_1). b) Variability of number of proposition facts in the final state of the test plan (s_1) in relation to the number of proposition facts in the final state of the reference plan (s_2). c) Adaptation changes (κ) evenly distributed between insertions (μ), deletions (ν) and permutations (ω) of ground actions over the 500 sample data set.

Finally, Plan Proximity and Semantic Plan Proximity are calculated over the plan-state pairs $[\pi_1, s_1]$ and $[\pi_2, s_2]$. The Change-Action ratio introduced in Section 3.4 is calculated at the end of the adaptation process. An additional measurement is also contemplated. This measurement determines the semantic impact that the new changes are providing to the adapted plan. This measurement calculates the distance to the closest ground action in the plan using $\hat{D}_g$ and its position distance $\hat{D}_l$ in relation to the point where the change is occurring. This is named the Closest Action-Action ratio:

$$\textit{Closest Action–Action ratio} = \frac{\sum_{i=0}^{\kappa} \left[\min_{j \in \pi_i} \left[\frac{\hat{D}_g(g_i, g_j) + \hat{D}_l(g_i, g_j, \pi_i)}{2} \right] \right]}{n_1 + \mu} \tag{3.33}$$

where π_i represents the plan after performing change i during the adaptation process. The position distance $\hat{D}_l$ is calculated as:

$$\hat{D}_l(g_1, g_2, \pi) = \frac{abs(pos(g_1, \pi) - pos(g_2, \pi))}{n_\pi} \tag{3.34}$$

where $pos(g, \pi)$ indicates the position of the ground action g in the plan π.

This process was repeated approximately 500 times.

Figure 3.6.a shows the variability of plan length between the reference plan length n_1 and the test plan length n_2. In this experiment, the number of effects of the actions was reduced considerably in relation to the experiment shown in Section 3.4.

This produced less variability in the amount of proposition facts at the final states (see Figure 3.6.b) than the one obtained during the experiment performed in Section 3.4. Figure 3.6.c shows how the changes introduced during the adaptation experiment were also evenly distributed between insertions, deletions and permutations of ground actions.

Table 3.5 shows the Spearman's ρ rank correlation coefficient for Plan Proximity and Semantic Plan Proximity in relation to the Change-Action ratio and the Closest Action-Action ratio. Figure 3.7 shows the distribution of Plan Proximity and Semantic Plan Proximity over the Change-Action ratio and the Closest Action-Action ratio.

In general, Semantic Plan Proximity shows higher values than Plan Proximity for the same data sets. This indicates that the compared plans are semantically closer that originally measured by the Plan Proximity metric. The results also show that, even after adding all the semantic interpretation to the metric, Semantic Plan Proximity correlates almost as well as Plan Proximity to the syntactic adaptation measurements calculated during the adaptation process. Finally, it can be seen that Semantic Plan Proximity is not affected by the measurement of the semantic impact of the changes ($\rho \sim 0.915$ for both adaptation ratios). On the other hand, Plan Proximity is affected by the inclusion of the semantic measurement in the adaptation process (dropping approximately 2.5 points in the correlation values).

These results lead us to conclude that the quantity of change is not as meaningful as the type of change made: Semantic Plan Proximity is more informed than Plan Proximity. Also, we have captured the semantic impact of the adaptation process, while at the same time, we have maintained the semantic metric independent from any domain-specific measurements. Ideally we would want human judgements to conclusively evaluate the advantage of Semantic Plan Proximity.

	Plan Proximity	Semantic *PP*
Change-Action ratio	-0.988	-0.916
Closest Action-Action ratio	-0.964	-0.915

Table 3.5: Spearman's ρ rank correlation coefficient for Plan Proximity and Semantic Plan Proximity in relation to the Change-Action ratio and the Closest Action-Action ratio.

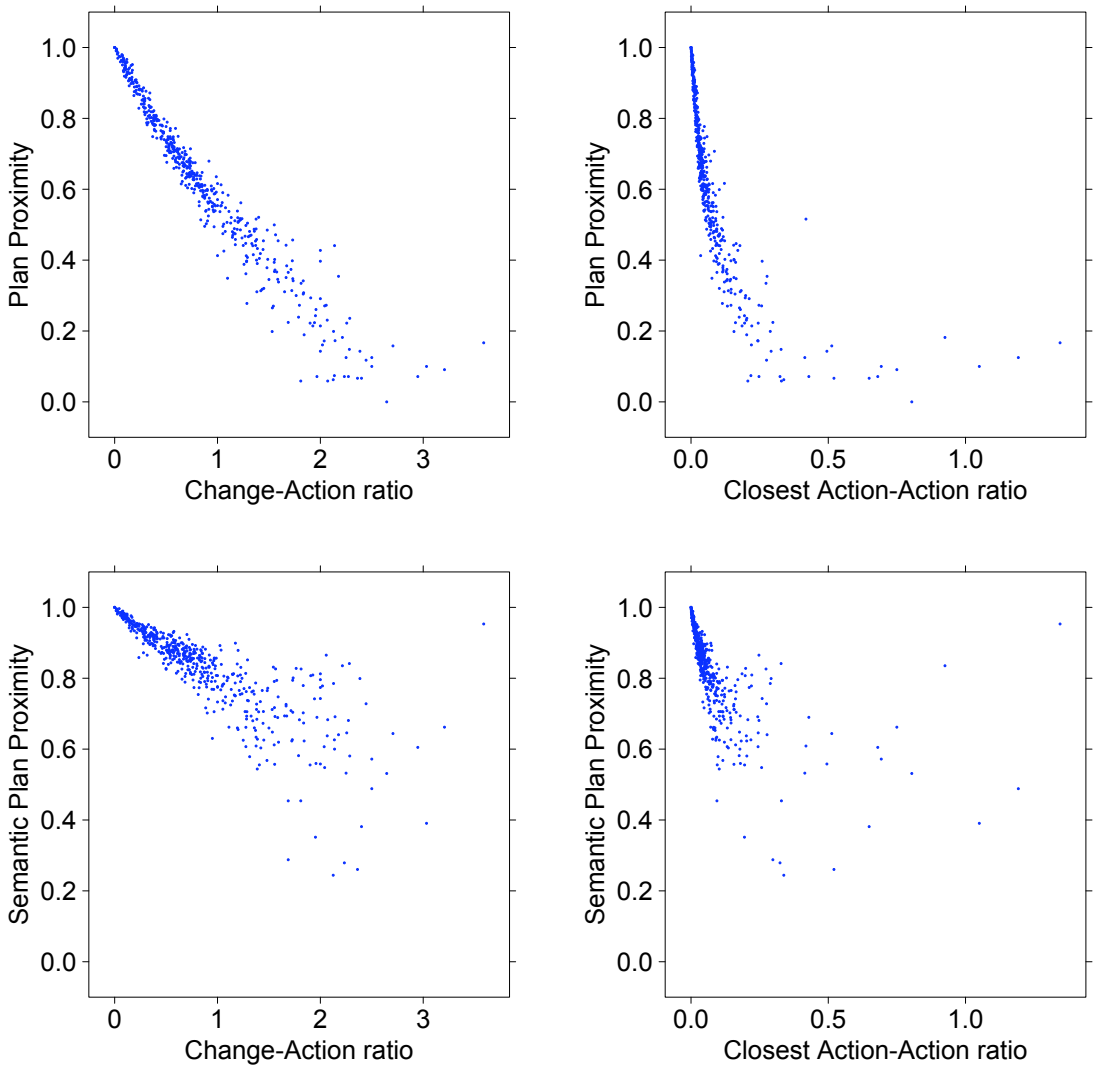


Figure 3.7: Distribution of Plan Proximity (top) and Semantic Plan Proximity (bottom) over the Change-Action ratio (left) and the Closest Action-Action ratio (right).

3.7 Summary and Outlook

In this chapter we presented a novel approach towards comparing different planning strategies solving the dynamic planning problem. The term 'Plan Proximity' is used to refer to a measure of the similarity between a reference plan and a test plan. This distance-based metric considers actions missing from the reference plan, extra actions added in the test plan, sequential ordering of the plans and the expected outcomes states of these plans. By taking all these factors into account, Plan Proximity showed to be more informed than Plan Stability in providing an enhanced view of the difference between plans. In conclusion, it is also more suitable for the comparison of

planning strategies in a dynamic environment. This metric is used in the next chapters to evaluate the different planning strategies proposed for solving the dynamic planning problem.

During the second half of this chapter, extended weights and mechanisms incorporate semantic knowledge in the metric formulation. By making use of two basic assumptions about the domain – class tree hierarchy and semantic separation of the propositions – the metric remains independent from any domain-specific measurements. As a consequence, Semantic Plan Proximity is able to capture, not only the quantity of change during the adaptation process, but also the semantic impact that these changes are introducing. By doing this, Semantic Plan Proximity is better informed than Plan Proximity. However, guaranteeing the total semantic separation of the propositions in the domain model is not trivial. As consequence, we predict that the more meaning overlap that exists between propositions for a given domain model the worse that the metric is going to perform. It will be necessary in a future extension of this work to undertake a human evaluation experiment in order to validate this metric with the views of a human expert.

3.8 Key Publications

- The specification of the Plan Proximity metric was presented in the paper *Plan proximity: an enhanced metric for plan stability* during the **Workshop on Verification and Validation of Planning and Scheduling Systems that took place during the 19th International Conference on Automated Planning and Scheduling** (ICAPS'09) in Thessaloniki (Greece) on September 2009 (Patrón and Birch, 2009).

Chapter 4

Continuous Mission Planning

Strive for continuous improvement, instead of perfection.

– Kim Collins

4.1 Introduction

We have highlighted the need for adaptive mission planning for autonomous decision making, in order to increase operability in UUVs. Adaptive mission planning required integrating orientation and observation in order to provide situation awareness of the mission environment. This integration was covered by the semantic-based framework and the set of ontologies presented in the second chapter. Then, the previous chapter presented methods capable of measuring the performance of adaptive mission planning strategies. These measures looked at the syntactic and semantic proximity between the original plan and the adapted plans.

In this chapter we propose a novel approach for adaptive mission planning for UUVs operating in a dynamic and uncertain discoverable mission environment. We assume that the information provided by the knowledge base is fully observable to the planner, i.e. the uncertainty arising from sensor limitations is handled by the agents processing lower-level data. We also assume that the mission environment is dynamic and uncertain, i.e. external events may occur and actions do not always perform as expected.

Under these assumptions, our approach implements a Bayesian paradigm for prediction, measurement, and correction inside a sequential decision-theoretic planning Markov decision process mathematical framework. Based on a continuous re-assessment

of the status of the mission environment, our approach provides a decision making loop capable of adapting mission plans. Instead of solving a plan from initial state to goals like in classical AI planning (Ghallab et al., 2004), it maintains a window of actions that it is believed can be performed from the current state in order to improve a given utility function.

Markov decision processes are normally implemented in robotics for motion planning in order to find optimal control policies that provide with the most suitable trajectories inside an navigation environment (Thrun et al., 2005). In our case, we implement an approach for mission planning. The goal of planning under this framework is to identify the policy that maximises the expected cumulative payoff. These policies are generally mappings from state to control actions. In our approach, we map from state to plan candidates. This means that we implement a policy that balances the selection of plans by using their estimated cost of execution and the reward obtained by reaching the new configuration of the mission environment.

We look at the cumulative payoff over a fixed window length. We adopt the value iteration model to implement the search policy, finding the best policy using classical state-space search. This combination of a finite approach with classical search allowed computational efficiency: by keeping track of the planned actions, we are able to forecast the impact of sensed events in the precomputed plan and, if necessary, react in advance. Given a planning horizon (window size), the solution provided by the approach is optimal for a greedy behaviour and pseudo-optimal for a lazy behaviour. Our implementation is able to handle temporal planning with durative actions, metric planning, opportunistic planning and dynamic planning.

Using the Plan Proximity metric presented in Chapter 3, the approach is evaluated under a static scenario and a partially known dynamic scenario. The comparison results show a high degree of similarity between our approach and the humanly driven adaptation. This indicates that the approach can be trusted by an operator as it provided similar outputs.

This chapter provides a solution to the main research objective of on-board real time autonomous adaptation for higher levels of decision making on UUVs (see Section 1.3.3.1).

4.2 Related Work

Space operations, Mars rovers in particular, have been the impetus behind autonomous planning and adaptability for unmanned vehicles. In this environment, the main motivation for mission plan adaptation is to reduce vehicle idle times and the handling of opportunistic events in order to maximise science return.

The *Remote Agent Experiment* (*RAX*) (Pell et al., 1997), which flew on Deep Space-1, demonstrated its ability to autonomously control an active spacecraft. This project highlighted the problem of using a batch planner on a reactive system. It took hours to replan after updates had invalidated the original plan.

Another system handling a certain level of planning adaptation in a real scenario was the *Automated Schedule and Planning Environment* (*ASPEN*) (Rabideau et al., 1999). This system classified constraint violations and attempted to repair each by performing a planning operation. However, the set of repair methods were predefined. Also, for any given conflict, its type determined the set of possible repair methods. The system could not guarantee that it would explore all possible combinations of plan modifications or that it would not retry unhelpful modifications.

Another system, called *CLEaR* (Fisher et al., 2000), used an extended version of the *ASPEN* system (Chien et al., 1999) in conjunction with a sequencer/controller. This combination provided a planning and execution framework for closed-loop commanding (Chien et al., 2000). The deliberative and reactive methods operated in parallel at run time to determine how to best respond to failures and take advantage of opportunities.

Mixed-initiative frameworks can also be found in the literature providing execution monitoring and continuous replanning (Myers, 1998) with incremental and collaborative scheduling (Wilkins et al., 2005) and continuous refinement of resource estimates (Morley et al., 2006).

In the underwater domain, the challenges for providing autonomous mission planning for UUVs were clearly stated by Turner (2005), Bellingham et al. (2006) and Patrón and Petillot (2008). This domain is now benefiting from the development effort of approaches previously validated in Space.

For example, the temporal constrained-based planner $EUROPA_2$ (Jónsson et al., 2000; Frank and Jónsson, 2003) used in *RAX* is now providing adaptive planning capabilities to oceanographers for maximising the science return of their AUV missions (Rajan et al., 2007; McGann et al., 2007). Using learning techniques for the

identification of features (Fox et al., 2007) and deliberative reactors for the concurrent integration of execution and planning (McGann et al., 2008a), live sensor data can be analysed during mission to adapt the control of the platform in order to measure dynamic and episodic phenomenon (McGann et al., 2008b, 2009). This architecture, known as *T-REX*, has been deployed successfully inside the Dorado AUV (Rajan et al., 2009).

In this chapter, we analyse a specific form of sequential decision-theoretic planning (LaValle, 2006). Unlike the previous aforementioned work, our approach is not domain, platform or mission specific. Based on a Markov decision process mathematical framework, it implements a Bayesian paradigm for prediction, measurement, and correction. These different phases nicely align with the stages of the OODA loop discussed in Section 1.4 that we identified are required to provide autonomous decision making for autonomous underwater vehicles. Thus, it makes it a suitable approach for our purposes.

4.3 Mission Environment

This section describes the concepts for the representation of the mission environment Π, including the representation of the domain model Σ and the problem model Ω. It is assumed that the planner has access to the knowledge describing the mission environment. This knowledge is extracted from the ontologies from the semantic-based knowledge framework described in Chapter 2.

4.3.1 Domain Model

The domain model is defined by the tuple $\Sigma = (C, O_C, V_C, P_V, A_V)$ presented in Section 2.4.2. This model contains the set of classes, objects, variables, predicates and actions of the domain. As mentioned in Chapter 2, the set of ground actions G_O and resources R_O can be directly calculated from this tuple.

A single extension to this model is introduced to contemplate the assumption that actions do not always perform as expected: the effects of an action can be probabilistic. Given an action a_h with probabilistic effects, the uncertainty matrix Γ for this action can be defined as:

$$\Gamma(a_h)_{|effect(a_h)|\times|R_O|} = \{p(i|j)| \; i \in effect(a_h), j \in R_O\} \tag{4.1}$$

where $p(i|j)$ is the probability of i given j.

4.3.2 Problem Model

A mission plan needs to be represented over time. $t \in \mathbb{R}$ defines the continuous time of the mission. $q \in \mathbb{N}_0$ defines a discrete step or slot in time of a certain duration $d_q : [t_0^q, t_n^q]$.

The problem model is defined by the tuple $\Omega = (x_q, \delta(O_C), Q_O, \gamma)$, where:

- x_q is the string representation of a state at the time slot q.
- $\delta(O_C)$ is the list of rewards of the set of objects O_C.
- Q_O is the set of proposition fact goals.
- γ is the cost function.

All these elements are described in detail in the following sections.

4.3.2.1 States

As described in Section 3.3.2.2, the approach uses a string representation x of a state s of the mission environment. A state at some particular step in time x_q is a set containing information i of the available actions, available resources and the combination of possible proposition facts at that step:

$$\begin{aligned} x_q = &\ x_q^{A_V} \cup x_q^{O_C} \cup x_q^{R_O} \\ = &\ \{i(\mathrm{x}) \in [0,1]\ \forall \mathrm{x} \in \{A_V \cup O_C \cup R_O\}\} \end{aligned} \tag{4.2}$$

It can be seen that $|x_q| = |A_V| + |O_C| + |R_O|$.

A ground action $g_q^{a_h}$ at step q defines a transition function between states $g_q^{a_h} : x_{q-1} \rightarrow x_q$ through the sequence of steps.

A plan candidate u_q^T defines a list of ground actions to be performed in the T steps ahead $u_q^T : x_{q-1} \rightarrow \langle g_q, g_{q+1}, \cdots, g_n | n \leq T \rangle$, where T defines the number of ground actions (window size) to be included in the continuous plan. T is also known as the *planning horizon*. The execution of g_q at time t is defined by e_q^t.

4.3.2.2 Rewards

Objects: Each object of the domain model $o_j^{c_i} \in O_C$ has a reward value $\delta(o_j^{c_i}) \in \mathbb{R}_+$.

Proposition facts: The reward of a proposition fact $r_y^{p_m} \in R_O$ is the sum of all the rewards of the objects used as arguments:

$$\delta(r_y^{p_m}) = \sum \delta(o_j^{c_i}) | \forall o_j^{c_i} \in arg(r_y^{p_m}) \tag{4.3}$$

The reward of a state is the sum of all the rewards of the proposition facts available in it:

$$\delta(x_q) = \sum \delta(r_y^{p_m}) i(r_y^{p_m}) | i(r_y^{p_m}) \in x_q^{R_O} \tag{4.4}$$

Mission goals: The set of mission goals can be defined explicitly with a list of direct rewards λ assigned to different proposition fact goals:

- $Q_O = \{\lambda(r_y^{p_m}) | r_y^{p_m} \in R_O, arg(r_y^{p_m}) \subseteq O_C\}$ is the set of proposition fact goals. The reward of the proposition fact defined by the operator is $\lambda(r_y^{p_m}) \in \mathbb{R}_+$. (e.g. `(= (surveyed seabedA) 100)`).

Ground actions: The reward of a ground action $g_q^{a_h}$ is related to the production of a proposition fact that has been explicitly defined as a mission goal in the mission problem. This means that the ground action has to produce a goal proposition fact in the new state that was not available in the previous state (see Eq. 4.5).

$$\begin{aligned} \delta(g_q^{a_h}, x_{q-1}) = \quad & \sum \lambda(r_y^{p_m}) \\ & | r_y^{p_m} \in \{x_q^{R_O} \cap Q_O\} \\ & \wedge r_y^{p_m} \notin \{x_q^{R_O} \cap x_{q-1}^{R_O}\} \end{aligned} \tag{4.5}$$

If a_h has probabilistic effects, the reward of the ground action $g_q^{a_h}$ is related to the probabilistic increase in the production of mission goals in the mission problem:

$$\begin{aligned} \delta(g_q^{a_h}, x_{q-1}) = \quad & \sum \lambda(r_y^{p_m}) \times \\ & (\Gamma(a_h)[r_y^{p_m} | x_{q-1}] - x_{q-1}^{R_O}[r_y^{p_m}]) \\ & | r_y^{p_m} \in Q_O \end{aligned} \tag{4.6}$$

4.3.2.3 Costs

Each ground action $g_q^{a_h}$ has an execution cost when being executed in a state. This cost is defined by $\gamma(g_q^{a_h}, x_{q-1}) \in \mathbb{R}$. This cost is related to specific domain metrics that are generally linked to the set of functions F_V of the domain model (e.g. `(energy ?component)`, `(duration ?start ?end)`, `(distance ?from ?to)`).

Note that, unlike rewards that can only be positive, the cost of an action can be also negative. For instance, they can produce value of the domain function instead of consuming it, e.g. energy.

4.3.2.4 Payoffs

The *payoff* or utility function of a ground action $g_q^{a_h}$ in a state x_{q-1} is defined by $\sigma(g_q^{a_h}, x_{q-1}) \in \mathbb{R}$. It represents the difference between the cost of the ground action and the rewards of the ground action and the generated state x_q:

$$\begin{aligned} \sigma(g_q^{a_h}, x_{q-1}) = \quad & \delta(x_q) \\ & +\delta(g_q^{a_h}, x_{q-1}) \\ & -\gamma(g_q^{a_h}, x_{q-1}) \end{aligned} \tag{4.7}$$

Hence, the cumulative payoff of a plan u_q of length T given a state x_{q-1} is the expected utility function of the plan at that state $\tilde{\sigma}(u_q^T, x_{q-1})$. This is calculated as the difference between the rewards accumulated by all the ground actions in the plan and all the expected states that they produce.

$$\tilde{\sigma}(u_q^T, x_{q-1}) = E\big[\sum_{\tau=0}^{T} \beta^\tau \sigma(g_{q+\tau}^{a_h}, x_{q+\tau-1})\big] \tag{4.8}$$

where $\beta \in [0;1]$ is known as the *discount factor*. This factor represents the fact that actions that are planned in the long term may have less effect over the current state than short term actions.

4.3.2.5 Passive action

An action should always exists called `passive-action` (ϕ). This action has no pre-conditions, no effects and a unitary cost ($\forall q, \gamma(g_q^{\phi}, x_{q-1}) = 1$). This action is used to generate stable plans in the continuous planning framework when the rest of the actions do not provide a positive payoff.

4.4 Semantic-based Continuous Mission Planning

The section presents a novel strategy for adaptive mission planning for UUVs. The description of the adaptive strategy and its evaluation were presented in Patrón et al. (2009a).

This approach is motivated by the need of a service-oriented architecture, a portable and extensible solution, a dynamic and uncertain environment, and the requirement to maximise operability during mission (see Section 1.3.3 for a full description of our research objectives). In order to develop a full decision making loop, this approach

implements a continuous iteration of observation, orientation, planning, and execution stages. The model of the mission environment is provided by the semantic-based knowledge framework described in Chapter 2. The observation of events, that dynamically change the mission environment, is combined with prior orientation and provided externally by other agents of the architecture. We assumed that these agents processing lower-level data are able to process the uncertainty arising from sensor limitations and to provide a fully observable knowledge to the planner (Reed et al., 2003, 2006b). As discussed, we also assume that the mission environment is dynamic and uncertain, i.e. external events may occur and actions do not always perform as expected.

Under these assumptions, our approach implements a Bayesian paradigm for prediction, measurement, and correction inside a Markov decision process mathematical framework. We apply this framework to an approach for mission planning. Instead of solving a mission plan from initial state to goals like in classical AI planning, we maintain a window of actions that it is believed can be performed from the current state in order to improve a given utility function. Once all actions from the window have been executed, the next iteration of mission planning is performed.

By keeping track of the requirements and expected effects of the actions planned ahead, we are able to forecast the impact of sensed events in the precomputed plan and, if necessary, react in advance. This reaction can be *greedy* or *lazy*. The greedy approach recalculates the plan as soon as the changes have been detected. Thus, it provides an optimal approach. Under a lazy behaviour, adaptation is delayed: planning is only performed at the end of the current window of execution. Thus, it provides a pseudo-optimal solution.

Figure 4.1 describes the workflow of the algorithm. This figure provides a high-level overview of the algorithm. At the top of the loop in this figure, the PLAN stage computes the best plan candidate π_q from a given mission environment Π_q (or the initial environment Π_0). The first ground action of the plan π_q is instantiated at the EXECUTE stage and executed in the functional layer of the platform via the ground action execution instance e_t^q. This stage produces a new predicted state $\tilde{x}_q$. This state is compared with the one provided by the knowledge base $\dot{x}_q$. Under a normal iteration, the state x_q is corrected based on this comparison at the CORRECT stage and the loop continues. This is the simplest form of the approach. However, a set of conditional cases can cause the approach to adapt to a new situation. These cases are captured in the figure by the conditional boxes: If the predicted state does not contain the state provided by the knowledge base, a new mission plan needs to be calculated. Additionally,

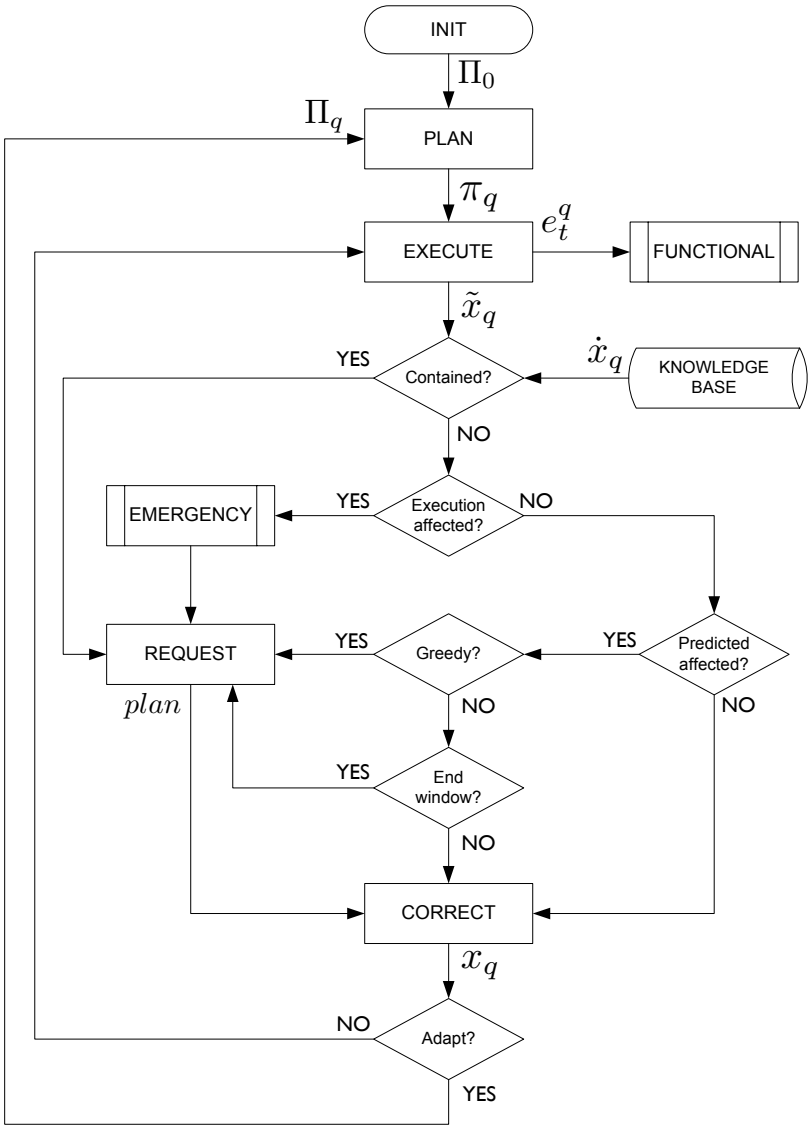


Figure 4.1: Workflow of the approach.

if the execution is affected by the state provided by the knowledge base, an emergency behaviour is necessary. A new plan also needs to be calculated if actions calculated ahead in the current plan are affected by the state provided by the knowledge base or if we have reached the end of the current plan. Under all these circumstances, an adaptation process is triggered sending the loop back to the PLAN stage.

The pseudo-code describing our approach can be found in Algorithm 4.4.1. Line (i) of Algorithm 4.4.1 indicates the beginning of the initialization of variables. Line 4.4.1.ii indicates the beginning of the continuous loop for adaptive mission planning for autonomous decision making in UUVs. This loop starts by searching for the plan

that maximises the utility function for the given plan horizon and current state (see Line 4.4.1.iii).

The goal of planning under this framework is to identify the policy that maximises the expected cumulative payoff. We implement a policy that maps from states to plan candidates $\pi : x \rightarrow u$. In this framework the current state is sufficient for determining the optimal control. A policy selects the plan candidate u_q^τ of horizon τ that maximises the expected cumulative payoff from the current state x_{q-1} (see Eq. 4.9).

$$\pi_q^\tau(x_{q-1}) = argmax_u \left[\sigma(u_q^\tau, x_{q-1})\right] | 1 \leq \tau \leq T \tag{4.9}$$

The best policy search is implemented using an exhaustive planning search in the state-space. This is represented in Algorithm 4.4.2. From Line 4.4.2. i, the best policy is initialised to the laziest mission plan, i.e. a sequence of passive actions. From Line 4.4.2. ii, the search variables are initialised. On Line 4.4.2. iii the call to the search function is performed.

The search recursive function is described in pseudo-code in Algorithm 4.4.3. On Line 4.4.3. i, if the plan horizon is reached the search is stopped. At this point, the new calculated plan is adopted if it provides a better utility function than the current best policy. If the planning horizon has not been reached, the set of possible ground action candidates U that can be executed from the current state is selected (see Line 4.4.3. ii). Finally, each ground action candidate is executed, its contribution computed, and the next recursive call of the function is performed (see Line 4.4.3. iii).

Once this is computed, Algorithm 4.4.2 returns the best policy found π_q^T, its total estimated payoff $\tilde{\sigma}(\pi_q^T, x_q - 1)$ and the plan matrix $\tilde{\Delta}_q^\tau$. Given a plan policy π_q^T, the *plan matrix* $\tilde{\Delta}_q^T$ contains information of the expected actions and resources used by the plan ahead. This matrix has T rows and $|A_V| + |O_C|$ columns.

$$\tilde{\Delta}_q^T = [\eta]_{T \times |A_V| + |O_C|} | \eta \in [0; 1] \tag{4.10}$$

Given a row $\varsigma \leq T$ and an action a_h,

$$\eta_{\varsigma, a_h} = \begin{cases} 1, \textit{ if } g_{q+\varsigma}^{a_h} \in \pi_q^T(x_{q-1}) \\ 0, \textit{ otherwise} \end{cases} \tag{4.11}$$

Given a row ς and an object o_j,

$$\eta_{\varsigma, o_j} = \begin{cases} 1, \textit{ if } o_j \in \tilde{x}_{q+\varsigma} \\ 0, \textit{ otherwise} \end{cases} \tag{4.12}$$

Algorithm 4.4.1: APPROACH(planning_horizon)

main

$t \leftarrow q \leftarrow 0$ (i)

$T \leftarrow$ planning_horizon

$x_q \leftarrow x_q^{A_V} \cup x_q^{O_C} \cup x_q^{R_O}$

greedy $\leftarrow$ **true** $\vee$ **false**

$q \leftarrow q+1$

forever (ii)

do {

$\tau \leftarrow T$

$(\pi_q^\tau, \tilde{\sigma}(\pi_q^\tau, x_{q-1}), \tilde{\Delta}_q^\tau) \leftarrow$ POLICY(x_{q-1}, T) (iii)

plan $\leftarrow$ **false**

while plan $\neq$ **true**

do {

$e_q^t \leftarrow g_0(\pi_q^\tau)$ (iv)

$(\tilde{x}_q, \tilde{\Delta}_q^{\tau-1}) \leftarrow exec(e_q^t)$ (v)

$\dot{x}_q \leftarrow$ knowledge_base (vi)

if $(|\tilde{x}_q| < |\dot{x}_q|)$ (vii)

then plan $\leftarrow$ **true**

else {

if $(\tilde{x}_q \supset \dot{x}_q) \wedge (\tilde{\Delta}_q^{\tau-1} \nsubseteq \dot{x}_q^{A_V} \cup \dot{x}_q^{O_C} | \forall \varsigma \leq \tau - 1)$

then {

if $(\tilde{\Delta}_q^{\tau-1}(e_q^t) \nsubseteq \dot{x}_q^{A_V} \cup \dot{x}_q^{O_C})$

then EMERGENCY() (viii)

plan $\leftarrow$ **true**

if $(\tilde{x}_q \subset \dot{x}_q) \wedge (\tilde{\Delta}_q^{\tau-1} \subseteq \dot{x}_q^{A_V} \cup \dot{x}_q^{O_C} | \forall \varsigma \leq \tau - 1) \wedge$ (greedy)

then plan $\leftarrow$ **true**

$x_q \leftarrow \dot{x}_q$ (ix)

$t \leftarrow t +$ duration(e_q^t)

$\tau \leftarrow \tau - 1$

$q \leftarrow q+1$

if $\tau = 0$ (x)

then plan $\leftarrow$ **true**

Algorithm 4.4.2: POLICY(x_{q-1}, T)

$\pi_q^T \leftarrow \{\phi_1, \phi_2, \cdots, \phi_T\}$ (i)
$\tilde{\sigma}(\pi_q^T, x_{q-1}) \leftarrow \texttt{goals}(x_{q-1}) + T \times \delta(x_{q-1})$
$\tilde{\Delta}_q^T \leftarrow \texttt{zero}_{T \times |A_V| + |O_C|}$
$\tau \leftarrow 0$ (ii)
$u_q^T \leftarrow \{\}$
$\tilde{\sigma}(u_q^T, x_{q-1}) \leftarrow 0$
$\tilde{\Delta}_q^\tau \leftarrow \texttt{zero}_{T \times |A_V| + |O_C|}$
SEARCH$(\pi_q^T, \tilde{\sigma}(\pi_q^T, x_{q-1}), \tilde{\Delta}_q^T, x_{q-1}, \tau, u_q^\tau, \tilde{\sigma}(u_q^\tau, x_{q-1}), \tilde{\Delta}_q^\tau)$ (iii)
return $(\pi_q^T, \widehat{\sigma}(\pi_q^T, x_{q-1}), \widehat{\Delta}_q^T)$

This matrix allows the prediction of the effects that events detected in the execution time line may have on the actions ahead on the mission plan. This prediction is managed by the conditional comparisons of estimated $\tilde{x}$ and observed $\dot{x}$ states found on Line 4.4.1.vii.

The search of the best policy on Line 4.4.1.iii corresponds to the Decision stage of the OODA-loop described in Section 1.4 of Chapter 1. Once this best policy is computed, the first action in the plan is executed (see Line 4.4.1.iv). This corresponds to the Action stage of the OODA-loop described in Section 1.4 of Chapter 1. Line 4.4.1.v predicts the new state of the environment based on the effects of the action being executed. This corresponds to the Orientation stage of the OODA-loop. Line 4.4.1.vi compares the predicted state with the observations coming from the knowledge base. This section corresponds to the Observation stage of the OODA-loop.

The conditions on Line 4.4.1.vii verify if a recalculation of the window plan is needed. On Line 4.4.1.ix the current state is updated to the observed state. Line 4.4.1.x asks for the calculation of a new window plan when the end of the window has been reached.

The details of the management of events captured by the conditions on Line 4.4.1. vii are described in the following sections.

Algorithm 4.4.3: SEARCH$(\pi_q^T, \tilde{\sigma}(\pi_q^T, x_{q-1}), \tilde{\Delta}_q^T, \tilde{x}_{q+\tau-1}, \tau, u_q^T, \tilde{\sigma}(u_q^T, x_{q-1}), \tilde{\Delta}_q^\tau)$

if $\tau = T$ (i)

then
- **if** $\tilde{\sigma}(u_q^T, x_{q-1}) > \tilde{\sigma}(\pi_q^T, x_{q-1})$
  - **then**
    - $\pi_q^T \leftarrow u_q^T$
    - $\tilde{\sigma}(\pi_q^T, x_{q-1}) \leftarrow \tilde{\sigma}(u^T, x_{q-1})$
    - $\tilde{\Delta}_q^T \leftarrow \tilde{\Delta}_q^\tau$
- **return**

$U_\tau \leftarrow \{g_{q+\tau} \in G_O \wedge \texttt{executable}(\tilde{x}_{q+\tau-1})\}$ (ii)

for each $e_{q+\tau} \in U_\tau$ (iii)

do
- $U_\tau \leftarrow U_\tau \setminus e_{q+\tau}$
- $(\tilde{x}_{q+\tau}, \tilde{\Delta}_q^\tau) \leftarrow \texttt{execute}(e_{q+\tau}, \tilde{x}_{q+\tau-1})$
- $u_q^T \leftarrow u_q^T \cup \{e_{q+\tau}\}$
- $\tilde{\sigma}(u^T, x_{q-1}) \leftarrow \tilde{\sigma}(u^T, x_{q-1}) + \beta^\tau \sigma(e_{q+\tau}, \tilde{x}_{q+\tau-1})$
- SEARCH$(\pi_q^T, \tilde{\sigma}(\pi_q^T, x_{q-1}), \tilde{\Delta}_q^T, \tilde{x}_{q+\tau}, \tau+1, u_q^T, \tilde{\sigma}(u^T, x_{q-1}), \tilde{\Delta}_q^\tau)$
- $u_q^T \leftarrow u_q^T \setminus \{e_{q+\tau}\}$
- $\tilde{\sigma}(u_q^T, x_{q-1}) \leftarrow \tilde{\sigma}(u_q^T, x_{q-1}) - \beta^\tau \sigma(e_{q+\tau}, x_{q+\tau-1})$

return

4.4.1 Action Management

There are four possible events that can be reported for an action. This section describes how the approach handles them differently.

a) *New action*: When a new action or capability is inserted into the system during the mission execution, the state model and the mission plan need to be corrected. In this process, the state is corrected by adopting the status provided by the knowledge base (see Line 4.4.1. ix). Similarly, the mission plan is corrected by calling for a new planning iteration (see Line 4.4.1. vii).

b) *Action recovery*: This case considers the recovery of an existent action that was temporarily unavailable. Under this situation, only the state needs to be corrected. At this point, the mission plan is not guaranteed to be optimal for the window of execution if a lazy approach is used. A lazy approach means that

the mission plan only gets recalculated once the current plan window has been executed. The greedy approach will also recalculate the mission plan to ensure that the action is being considered.

c) *Removal of an action that is not in plan*: In this case, the state is corrected and the execution continues.

d) *Removal of an action that is used in the plan*: In this case, the state and the mission plan need to be corrected. The identification of actions needed by the mission plan ahead is performed by looking at the plan matrix.

4.4.2 Object Management

Similarly, this section describes how the approach handles events related to objects of the domain. There are four types of events for objects:

a) *New object*: When a new object is inserted in the knowledge base, the state model and the mission plan need to be corrected. This situation follows a similar process as per the insertion of a new action.

b) *Object recovery*: In a similar way to the action recovery, the state should be corrected when an existent object that was temporarily unavailable is recovered. Under this situation, the plan is not guaranteed to be optimal for the window of execution if a lazy approach is used. The greedy approach recalculates the mission plan to ensure that the object recovered is being considered.

c) *Removal of an object that is part of the current state*: This is the only situation that can cause an unstable condition in the decision making loop. Since, an object used by the action being executed is reported as unavailable by the knowledge base. This situation is handled by the call to the EMERGENCY behaviour on Line 4.4.1. viii. This behaviour is domain dependent. In the unlikely situation of this occurrence, this behaviour provides a robust controlled recovery of the platform.

d) *Removal of an object used in the plan*: In this case, the state needs to be corrected and the mission plan recalculated.

e) *Removal of other objects*: In this case, only the state needs correction.

4.4.3 Proposition Management or Explicit Goals

Propositions can be managed through the use of goals. Goals are set of proposition facts that the operator want to happen. Goals are described explicitly by the operator in Q_O (see Section 4.3.2.2).

4.4.4 Action with Probabilistic Effects

As discussed in Section 4.3.1, actions can have probabilistic effects. In this case, the information values i of the state vector x_q are probabilistic. The reward of the proposition facts in the goals is affected by the probabilistic effect of the actions. The probabilistic effect of each of the actions is represented by $\Gamma(a_h)_{|effect(a_h)|\times|R_O|}$. As described in Eq. 4.6, the effects of these actions influence the calculation of the payoff function of the related ground actions.

This type of actions are very relevant for Reactive Data Gathering (RDG) applications (see Section 1.3.2.2). Classification algorithms for automatic target recognition (ATR) provide confusion matrices with probabilistic information (Pailhas et al., 2010). This information can be mapped into the uncertainty matrix. These processes guide the estimated reward of the re-observations of targets. This interaction between ATR algorithms and the adaptive planning approach is currently being studied under the projects described in Section 7.4.

4.4.5 Information Exchange

It can be seen that information about the current availability of actions and resources is stored in a single binary state vector. This vector can be easily compressed and transferred using low bandwidth communication hardware such as acoustic modems.

4.5 Evaluation of Continuous Mission Planning

Using the Plan Proximity metric presented in Chapter 3, the approach has been evaluated under the scenario described by the Student Autonomous Underwater Challenge - Europe mission rules (SAUCE, 2009). For this competition, the mission environment is comprised of three aligned gates, a hovering bottom target, a moving mid-water target, two wall sections and a docking station. The scenario is represented in Figure 4.2.

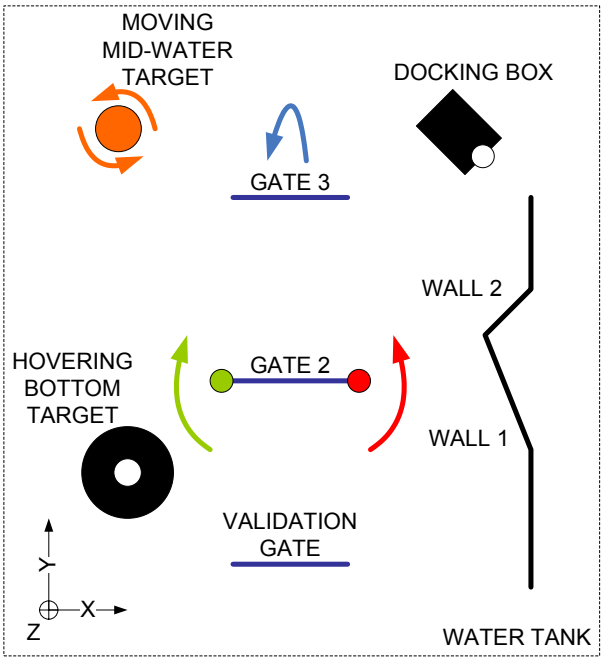


Figure 4.2: Scenario of the 2009 Student Autonomous Underwater Challenge - Europe.

In this challenge, the vehicle should pass through three aligned gates (`gate1`, `gate2`, and `gate3`), first forward and then backwards. Green and red lights on the second gate – objects `green`, `red`, and `off` – indicate the route that should be taken for its avoidance during the forward pass: `left`, `right` or `middle` of the gate respectively. After passing the gates in and out, the vehicle should attempt (in no particular order) to perform an inspection of the hovering bottom target (`bottom`), to follow the moving mid-water target (`middle`), to survey the wall sections (`wall1` and `wall2`) and to dock at the docking station (`recovery`).

Even if this scenario may not look very complex on AI terms, it challenges state of art autonomous capabilities of current underwater platforms. For evaluation purposes, the scenario is described using four files based on the PDDL syntax (Ghallab et al., 1998):

- The *Domain* file $\mathcal{D}$: describes the domain model Σ, including the classes in C, constants of O_C, the predicates P_V, the functions F_V and the actions A_V. For this domain, the action `toWait` is identified as the passive action ϕ.
- The *Problem* file $\mathcal{P}$: describes the problem model Ω, including the initial state x_0, the rewards $\delta(O_C)$, the goals Q_O and the cost metric γ.
- The *Environment* file $\mathcal{W}$: describes the known objects in O_C and their particular

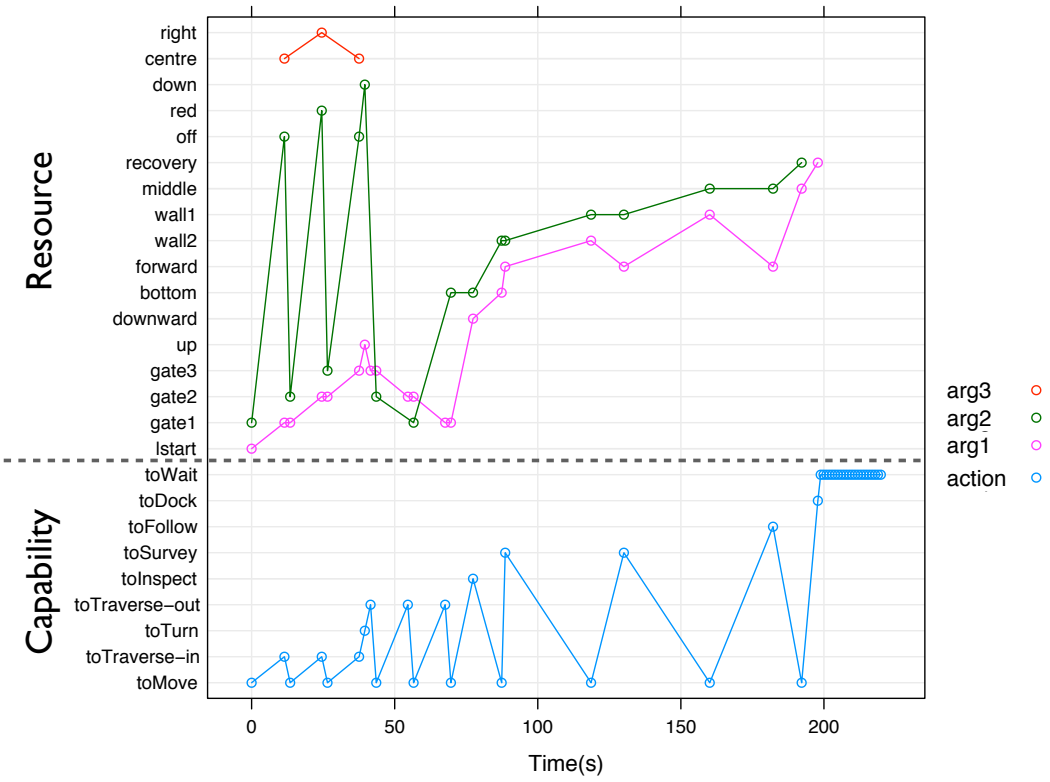


Figure 4.3: Human generated reference plan π_0 with actions evolving over time. This plan is used as ground truth for the evaluation of the different planning strategies. The dotplot represents instances of actions with their arguments at the point in time where they are executed (e.g. action `toMove` with first argument (arg1) `lstart` and second argument (arg2) `gate1` create the ground action instance `(toMove lstart gate1)` that is executed at $t = 0$).

domain properties. These properties allow the calculation of the function values of F_V.

- The *Dynamic environment* file $\mathcal{Y}$: simulates events that occur on the execution time line. Events can be triggered by a predicate in the current state or by reaching a particular time slot. They can add, delete and restore objects, actions and predicates from the current knowledge of the mission environment.

4.5.1 Known Static Environment

This section analyses the outcomes of the proposed adaptive planning strategy in a fully known static mission environment. For this scenario, the environment file contains all

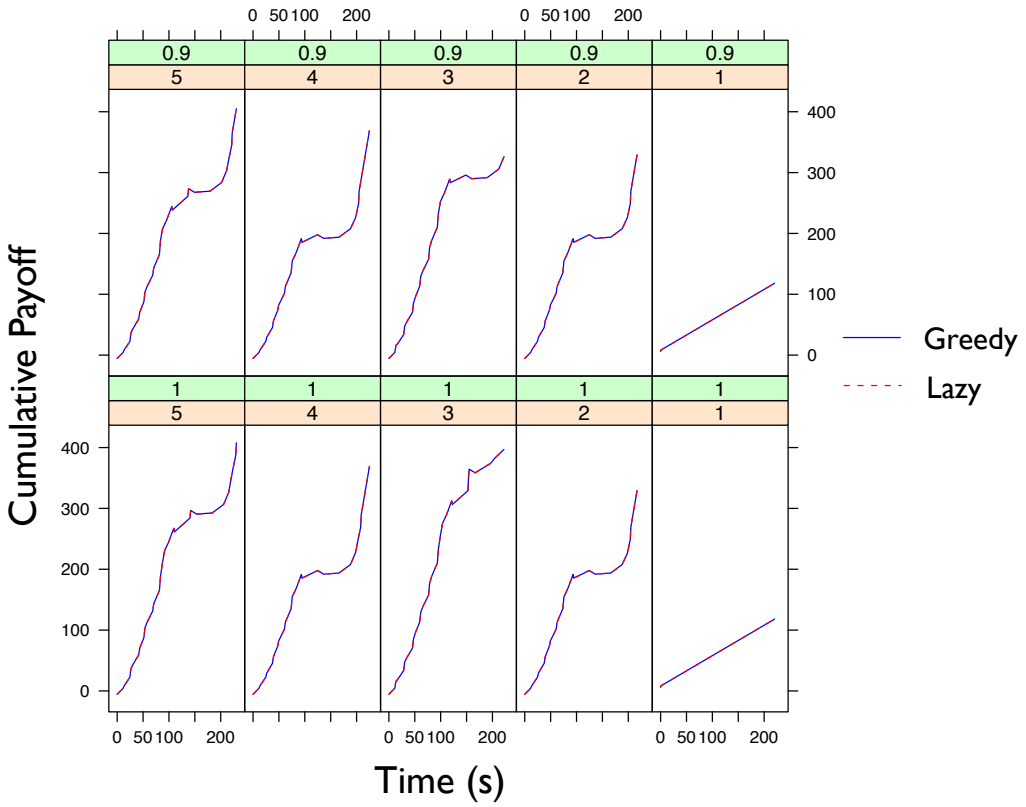


Figure 4.4: Plot matrix for of the cumulative payoff of the planning strategy solving the known static scenario. Columns represent the different values of the planning horizon $T \in [1;5]$. Rows represent the different discount factor values $\beta \in [0.9;1]$. The two lazy factors $lazy \in [0;1]$ are captured by the two entry lines.

the relevant information about the environment and the dynamic environment file is empty.

The human generated mission plan, named π_0, is used as reference ground truth for the evaluation of the different planning strategies. This mission plan is represented in Figure 4.3. The figure shows the ground actions at the point in time where they are executed. Ground actions are formed by the action and the list of objects that are used as arguments.

The proposed adaptive planning strategy is executed with different values of the planning horizon T, the discount factor β and the laziness factor. The different outcomes are evaluated by looking at their Plan Proximity to π_0. As discussed, instances of the passive action ϕ provide continuity in the decision making process. But, they do not affect the outcome of the adaptation process. As a consequence, all instances of the passive action ϕ that may appear on any of the plans are removed before perform-

T	1	2	3	4	5
$\hat{D}_p$	1.00	0.00	0.00	0.00	0.00
$\hat{D}_s$	0.21	0.00	0.00	0.00	0.00
$PP_{0.5}$	0.39	1.00	1.00	1.00	1.00

Table 4.1: Normalized plan distance ($\hat{D}_p$), normalized state distance ($\hat{D}_s$) and Plan Proximity ($PP_{0.5}$) to π_0 for the approach using $T \in [1;5]$ and $\beta = 1$ in the known static environment.

ing their comparison. All the strategies, including the human, were given a maximum execution time of $t = 220$ seconds.

Figure 4.4 shows the cumulative payoff for the different parameters used in the adaptive planning strategy. Table 4.1 shows Plan Proximity to π_0 of the different planning strategies with $\beta = 1$. Looking at the figure and the table, it can be seen that the plan strategy using $T = 1$ obtains the lowest cumulative payoff and scores the lowest in the Plan Proximity to π_0. By having such a short planning horizon, this strategy does not have enough evidence about rewards to commit to different actions other than the passive action. Thus, by not having enough lookahead and by staying in a 'comfortable' configuration this strategy ends up providing the poorest result. On the other hand, planning strategies with $T \in \{2,3,4,5\}$ obtained Plan Proximity values to π_0 that indicate that these plan strategies generate the same plan as the human. Furthermore, they obtained a similar cumulative payoff at the end of their executions, with the higher planning horizons scoring the highest values. These higher scores came at a price, the further the planning strategies were asked to look ahead, the longer they took to compute the plans. Section 7.3.4 of the final Chapter 7 proposes an extension to this work to mitigate this computational issue.

The results show that the discount factor β does not have a significant effect in any of the strategies. This is because the planning horizons used are short in comparison with the penalty that the discount factor introduces.

Finally, a lazy or greedy execution does not have any impact on the final results. This is because in the static scenario the mission environment is known a priori and does not change over time. This will change during the execution under a partially-known dynamic environment.

T	H	1	2	3	4	5
$\hat{D}_p$	0.18	1.00	0.41	0.15	0.20	0.20
$\hat{D}_s$	0.01	0.25	0.06	0.01	0.01	0.01
$PP_{0.5}$	0.90	0.37	0.76	0.91	0.89	0.89

Table 4.2: Normalized plan distance ($\hat{D}_p$), normalized state distance ($\hat{D}_s$) and Plan Proximity ($PP_{0.5}$) to π_0, the humanly driven mission for the static environment. Results in the partially-known dynamic environment of the humanly driven mission H and the different approach strategies obtained using $T \in [1;5]$ and $\beta = 1$.

4.5.2 Partially-known Dynamic Environment

This section analyses the outcomes of the proposed adaptive planning strategy when solving the dynamic planning problem. This experiment uses the same scenario as in the previous section (see Figure 4.2). However, in this case the mission environment is not fully known *a priori*. But instead, it is discovered through the execution of the mission.

In order to simulate this, the environment file only contains information about the first gate (`gate1`) and the docking station (`recovery`). This is the only knowledge about the environment available before starting the mission. The dynamic environment file simulates a series of events that add, delete or restore objects and/or actions to the mission environment model.

Figure 4.5 shows the evolution of capabilities and resources over time for the case of the humanly driven mission. It can be seen how resources are incorporated to the knowledge base as they are discovered. Also, some of the resources and actions are unavailable for finite periods of time limiting the capabilities of the platform.

Figure 4.6 represents the humanly driven mission execution H. This execution is the result of on-line adaptation of the mission performed by the human. This adaptation process copes with the changes in the mission environment perceived by the updates on the knowledge base. Both humanly-driven missions (static and dynamic versions) achieve all the mission objectives for this scenario. However, the sequence of ground actions performed is different. This can be seen in the different action sequences displayed by Figure 4.3 and Figure 4.6. This is due to the fact that under the dynamic scenario the operator does not hold a full picture of the environment from the start of the mission. For instance, it can be seen how the operator is forced to insert a series of passive action instances while the action `toMove` stays unavailable while being at

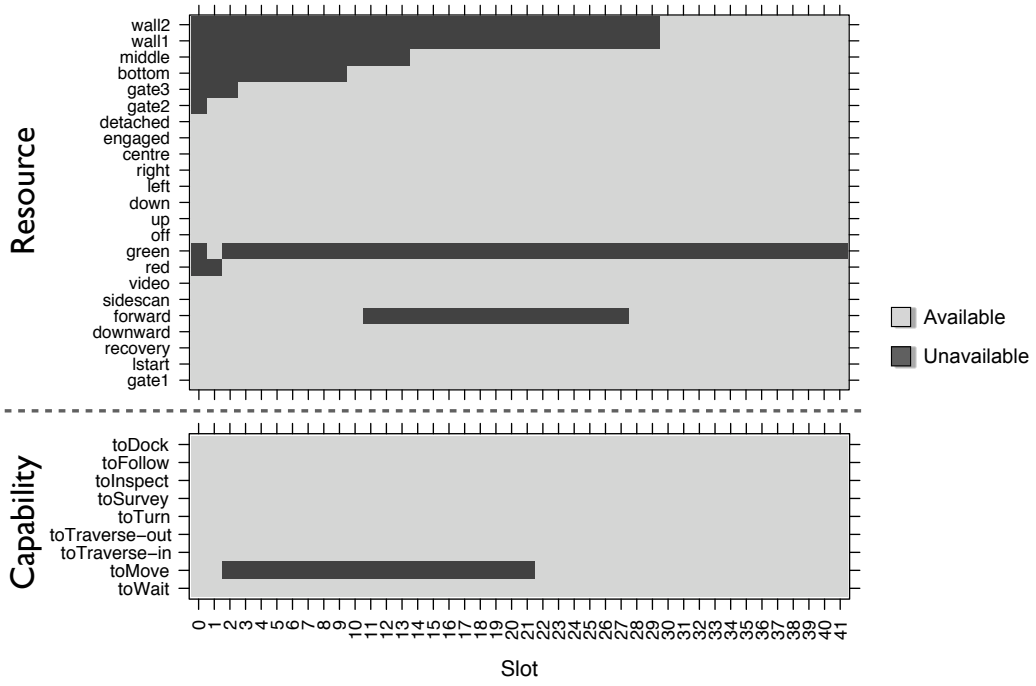


Figure 4.5: Evolution of capabilities and resources over time slots for the humanly driven mission (dark grey means unavailable). The resources `gate2`, `gate3`, `bottom`, `middle`, `wall1` and `wall2` are discovered during the mission. The lights at `gate2` (see `red` and `greeen`) are discovered and change colour during the mission. The resource `forward` camera and the capability `toMove` become temporarily unavailable during the mission.

`gate2` until the time slot $q = 22$.

Table 4.2 represents the Plan Proximity to π_0 of the planning strategies, including the human, solving the dynamic planning problem scenario. Figure 4.7 shows the cumulative payoff for the different planning strategies.

The Plan Proximity results indicate that the planning strategy with $T \in \{3,4,5\}$ obtains solutions that are as close to π_0 as the human solution H. Also, the lazy approach generally lags behind the greedy approach in accumulating similar payoff values in strategies with a high plan horizon, e.g. $T = 5$.

As for the static results, the discount factor β does not have a significant impact in the strategies. Similarly, myopic strategies, i.e. $T \leq 2$, tend to find comfortable configurations that provide them with high rewards without obtaining the expected mission objectives. This effect can be mitigated by training the algorithm and mission objective parameters – factors, rewards, and costs – using a large data set of humanly generated missions. This analysis is proposed in Section 7.3.3 of Chapter 7 as a possible extension of work.

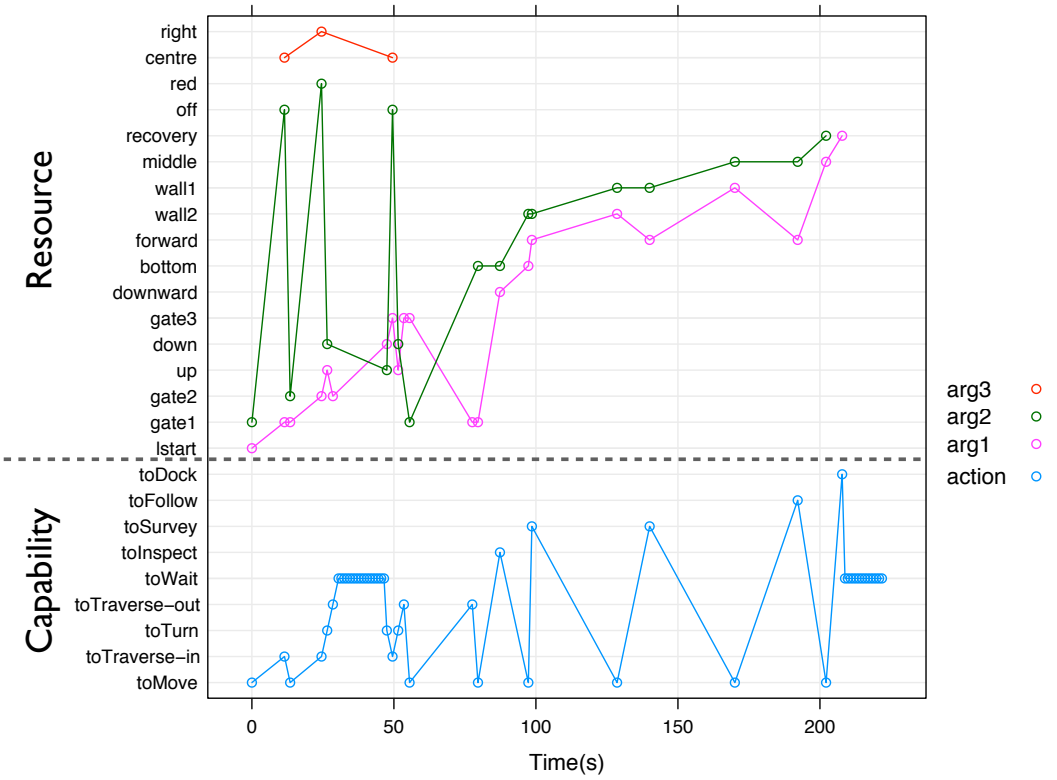


Figure 4.6: Human driven mission for the partially-known dynamic environment scenario.

4.6 Summary and Outlook

In this chapter we presented a novel approach for adaptive mission planning specifically suited to UUVs operating in dynamic and uncertain discoverable mission environment. Based on a continuous re-assessment of the mission environment, our approach provides a decision making loop capable of adapting mission plans. It implements a Bayes paradigm for prediction, measurement, and correction inside a Markov decision process mathematical framework. The goal of planning under this framework is to identify the policy that maximises the expected cumulative payoff. We implement a policy that balances the selection of plan candidates by using their estimated cost of execution and the reward obtained by reaching the new configuration of the mission environment. By combining a finite value iteration approach with a classical search of the best policy, we are able to efficiently forecast the impact of sensed events in the pre-computed plan and, if necessary, react accordingly. Given a planning horizon (window size), the solution provided by the approach is optimal for a greedy behaviour and pseudo-optimal for a lazy behaviour. Our implementation is able to handle temporal

planning with durative actions, metric planning, opportunistic planning and dynamic planning.

Using the Plan Proximity metric presented in Chapter 3, the approach was evaluated under a static scenario and a partially known dynamic scenario. The comparison results showed a high degree of similarity between our approach and the humanly driven adaptation.

Future extensions aim to mitigate the current computational limitations of extending the planning horizon, to incorporate the expected probabilistic effects of actions provided by other agents, and to extract knowledge from the human expert to obtain more informed policies.

This chapter highlighted how actions can be dynamically incorporated to and removed from the list of available capabilities of a platform. The next chapter focuses on adaptive planning strategies that cope with the dynamic discovery of system capabilities inside a service-oriented framework.

4.7 Key Publications

- The results of this chapter were presented in the paper *Continuous mission plan adaptation for autonomous vehicles: balancing effort and reward* during the **4th Workshop on Planning and Plan Execution for Real-World Systems during the 19th International Conference on Automated Planning and Scheduling** (ICAPS'09) that took place in Thessaloniki (Greece) on September 2009 (Patrón et al., 2009a).

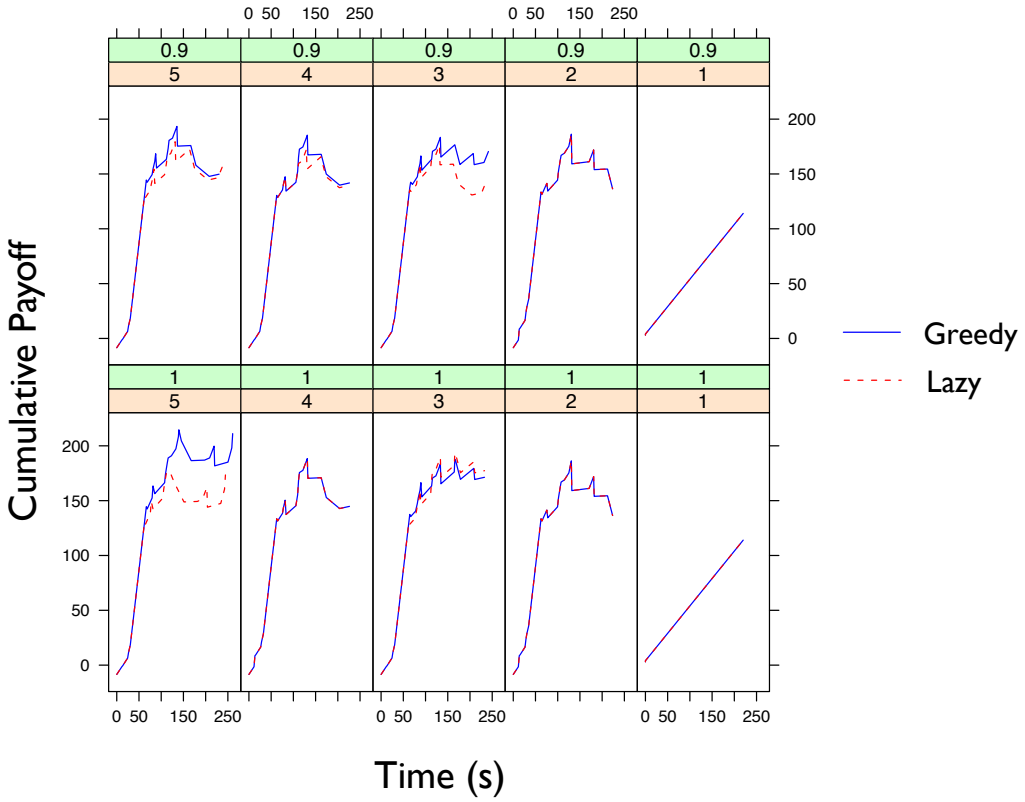


Figure 4.7: Plot matrix for of the cumulative payoff of the planning strategy solving the partially-known dynamic scenario. Columns represent the different values of the planning horizon $T \in [1;5]$. Rows represent the different discount factor values $\beta \in [0.9;1]$. The two lazy factors $lazy \in [0;1]$ are captured by the two line entries.

Chapter 5

Service-Oriented Mission Planning

If you have knowledge, let others light their candles in it.

– Margaret Fuller

5.1 Introduction

In recent years, maritime disciplines requiring the use of underwater robots have come to realise that the success of the mission does not depend primarily on their hardware platform. The software capabilities of the payloads and the instruments available to them, have become increasingly important.

It is for this reason that underwater robotics is now slowly moving toward service-oriented and open architectures. Under such framework, payloads on the platforms are physically installed, reconfigured, and removed, sometimes even during the short maintenance process carried out in between missions. In some cases, payloads can also be enabled or disabled during the mission itself. These payloads are providing a wide range of independent capabilities and loosely coupled services. In order to accommodate this type of reconfigurable framework, platform manufactures are starting to integrate modular payload bays that facilitate their integration in the system. Examples of this modularity can now be found in platforms such as the GAVIA AUV (Thorhallsson and Hardason, 2003) and the Ranger RN AUV (Schulz et al., 2005).

The problem is that these payload changes need to be manually published to the rest of the system. In order to provide mission flexibility and to maintain the platform's operability, it is important that the operator is released from the tedious task of having

to communicate these changes to the other intelligent agents. Recent technologies are helping to overcome this issue. They allow the automatic publication of these capabilities as they become available in the system.

The focus of this chapter is to adapt mission plans based on the dynamic discovery of system capabilities that are not known *a priori*. Our approach finds mission plans that cover the mission requirements and that agree with the discovered capabilities provided by the modular technologies that have been mentioned previously. We show how independent service-oriented agents (SOA) coordinate with a goal-based planner using the situation-aware knowledge-based framework. These two elements provide the interoperability of embedded intelligent agents for distributed service discovery and embedded decision making. Agents might be distributed among the different payloads working in collaboration inside a platform. Service-oriented mission planning makes platforms more accessible to human operators.

The approach is evaluated under a service discoverable scenario using a series of communication messages to publish and report the capabilities of the available agents. Results show that this discovery-based implementation finds the same results as the baseline which was explicitly provided with the platform configuration.

This chapter provides a solution to the research objective of service-based decision making for adaptive mission planning (see Section 1.3.3.3).

5.2 Service-Oriented Mission Environment

Before introducing the approach to service-oriented mission planning for UUVs, it is necessary to define how service-oriented architectures fit inside the new UUV platform designs. The core principle of services and service-oriented architectures is the design of systems by means of distributed capabilities that are self contained, loosely coupled, and have well defined interfaces.

The definitions introduced in this section are an extension to the mission environment model described in Section 4.3. Also, they include some of the system components defined by the JAUS architecture (SAE-AS-4 AIR5665, 2008).

In this sense, a UUV platform can be seen as a distributed system (see Section 1.4). Physically, a UUV platform can be made of several payloads.

Definition 5.2.1 *A payload is an individual hardware entity in the platform providing a collection of resources and agents to the unmanned system.*

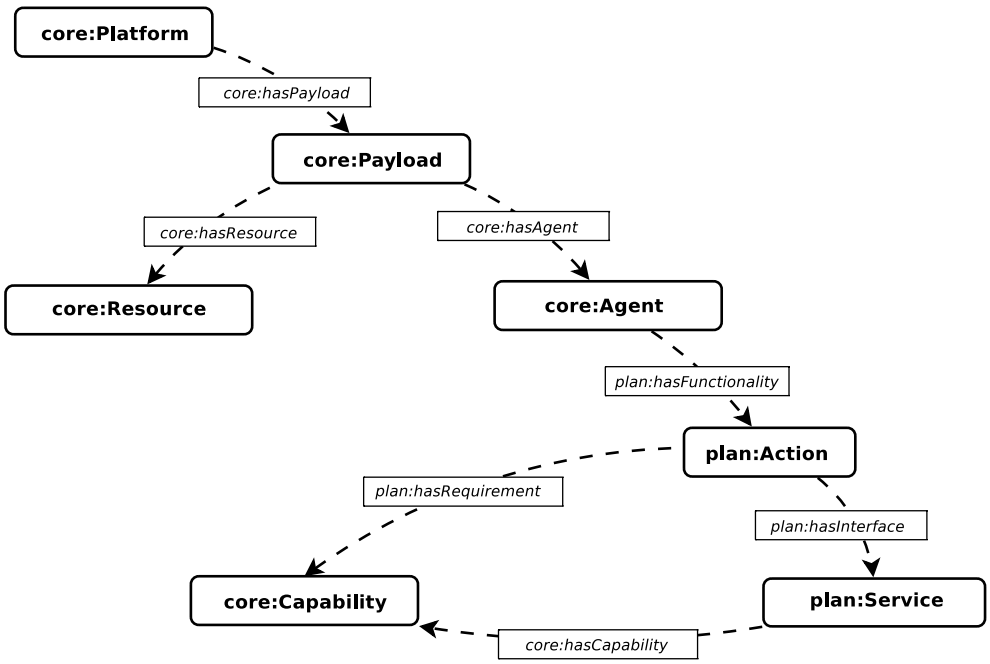


Figure 5.1: Concepts and axioms for service discovery and mission planning in a service-oriented architecture.

A UUV platform aggregates the power of these entities to collaboratively run a mission plan in a transparent and coherent way, so that they appear as a single individual entity.

Definition 5.2.2 *A resource is a physical component or instrument of the system.*

Each system resource belongs to a certain class and it is modelled as an object of that class ($o_j^{c_i} \in O_c$).

Definition 5.2.3 *An agent is an individual unit of processing that implements a specific functionality in the system.*

A function performed by an agent is modelled by an action ($a_h \in A_v$), and can be parametrised and instantiated by a ground action ($g_z^{a_h} \in G_O$).

In order to perform its function, each agent has a set of requirements ($\texttt{condition}(a_h)$). By performing a function, each agent provides a set of effects over the resources of the system ($\texttt{effect}(a_h)$). Thus, an agent is at the same time consumer and producer of services.

Definition 5.2.4 *A service is considered as the interface to a functionality.*

A service specifies only the messaging syntax, semantics, and protocol necessary to use the function. It does not specify how the function is implemented. A service is

a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by the agents.

Definition 5.2.5 *A capability is each of the services produced by an agent when performing its action.*

Definition 5.2.6 *A requirement is each of the services consumed by an agent to perform its action.*

A snapshot of the ontology concepts capturing this definitions is presented in Figure 5.1.

The human operator can be considered as one of the agents in the system. In this case, the mission goals can be seen as the set of requirements for that agent. A goal is modelled by a set of proposition facts ($r_y^{p_m} \in Q_o$).

5.3 Approach

The first part of this section describes how the adaptive mission planning agent can dynamically gain access to the services available in the system. The second part describes the novel approach proposed to service-oriented mission planning.

5.3.1 Related Work

Many are the steps required for a payload to join a UUV platform:

- Plug device into a port of the platform,
- Install the device software, configuration files, and metadata,
- Modify the configuration of the host platform to recognize the device, and
- Note change of data collection context and associate metadata with new data stream.

These tasks all are time-consuming, tedious, and prone to errors when performed manually by an operator. They also do not scale well.

New technologies and protocols are appearing allowing the set of services of the payload to be automatically retrieved once the device has been physically plugged into a port of the host platform. In these protocols, metadata is automatically inserted into

the data stream. Thus, it becomes accessible to the rest of intelligent agents in the system.

An instance of these type of technologies is the Programmable Underwater Connector with Knowledge (PUCK) from the Monterey Bay Aquarium Research Institute (MBARI) (O'Reilly, 2009). The PUCK protocol is a simple command protocol that helps to automate the configuration process by physically storing information about the instrument with the instrument itself. PUCK-enabled payloads are able to perform an automatic configuration process and to provide a very flexible plug-and-work solution. PUCK is mainly available for serial ports.

Another example is the standard of the JAUS Service Interface Definition Language (JSIDL) (SAE-AS-4 AS5684, 2008). This protocol is based on TCP/IP ports. It is a top-level specification that defines a language for describing unmanned system capabilities. It uses a syntax as a means to formally specify a schema that is to be used to create JAUS Service Definitions (JSDs) for all defined capabilities.

Technologies such as the aforementioned provide the required machine readable information that enables the autonomous adaptation of mission plans to the capabilities available in the platform. Together, automatic service discovery and autonomous service-oriented mission adaptation can provide benefits for long-term deployment and easy reconfiguration of UUV platforms.

5.3.2 Service-oriented Mission Planning

In this section we present a different goal-based approach from the one presented in Chapter 4. It is based on backward state-space searching. The first part of this section describes how this approach autonomously generates a mission plan that covers the mission requirements using the knowledge available about the platform capabilities. The second part introduces an extension to this approach by proposing a service-based mission planning approach capable of generating a mission plan using the capabilities of the platform that are not known *a-priori* but dynamically discovered instead.

5.3.2.1 Autonomous mission planning

An instance of a mission environment for UUVs was defined in Section 1.4 as $\Pi = \{\Sigma, \Omega\}$, where Σ is the domain model containing a set propositions defining the available resources in the environment and the set of actions of the platform (see Section 4.3.1), and Ω is the problem model containing the initial state x_0 and Q_O is the mis-

sion requirements or set of possible mission accomplished states (see Section 4.3.2).

A **progression** state-transition function $\varphi : G_O \times S \rightarrow S$ represents the expected transition between an original and a result state provided by the execution of a ground action in the original state. Given an instance Π and a progression state-transition function φ, the mission generation problem consists in finding if there exists a mission plan π_0^T of length T, using instances of the available actions $g_z^{a_h} \in G_O$, such that if applied to the current platform state x_0 satisfies the requirements in Q_O. The mission plan π_0^T can be seen as a sequence of ground actions through the progression function (see eq. 5.1).

$$\pi_0^T = \langle g_1, g_2, \ldots g_T \mid x_q = \varphi(g_q^{a_h}, x_{q-1}) \rangle \tag{5.1}$$

Several approaches exist in the artificial intelligence (AI) literature capable of solving this problem (Ghallab et al., 2004). If the set of mission requirements Q_O is considered to be provided by the operator, the process of backward state-space search fits the logical model for generating a mission plan that is coupled with the current system functionality. The method starts from the mission requirements given by the operator Q_O and applies the **regression** function $x_{q-1} = \varphi^{-1}(g_q^{a_h}, x_q)$ using instances of the available actions to produce new states, stopping if a set of sub-goals satisfied by the initial state x_0 is produced.

We implemented a backward chaining approach to be used as reference during the experiments carried out in Section 5.5. This implementation can deal with several extensions from the classical representation. It can handle typing, fluents, attached procedures, and durative actions. This concepts are modelled in the domain by C, V_C, F_V and A_V respectively. In the search process, it can also incorporate different heuristics and contemplate domain-specific metrics.

The mission plan is generated by calculating the estimated payoff function $\tilde{\sigma}$ of the visited states. The search process is guided towards the minimisation or maximisation of the values of this function (e.g. `(:metric minimize (+ (distance-travelled) (total-duration)))`, `(:metric maximize (battery))`, etc.).

Based on $\tilde{\sigma}$, a node-selection function $ns(W)$ is implemented to choose which state w from a set of candidate nodes W to visit next from a state x_q (see eq. 5.2). The use of *argmin* or *argmax* in eq. 5.2 is related to the selected metric optimisation. The function $\texttt{executable}^{-1}$ returns the list of ground actions that are able to make a regression step from the state x_q, i.e. actions which capabilities are all part of the state.

$$\begin{aligned} x_{q-1} &= ns(W) = [argmin|argmax]\{\tilde{\sigma}(g_q^{a_h}, w) \mid w \in W\} \\ W &= \bigcup_{g_q^{a_h} \in \{executable^{-1}(x_q) \cap A_V\}} \{\varphi^{-1}(g_q^{a_h}, x_q))\} \end{aligned} \tag{5.2}$$

For evaluation purposes, different heuristic functions h have been implemented that guide the node-selection step during the search process. They estimate the distance to the initial state x_0 from the current node x_q. They have been selected to have different admissible and informative properties. The heuristic h_a simply calculates the remaining resources in the node needed to match the current platform state resources (see eq. 5.3). The heuristic h_b calculates the sum of the cost of achieving all the requirements in x_0 by looking at the requirements ($\texttt{condition}(g_q^{a_h})$) and positive effects ($\texttt{effect}^+(g_q^{a_h})$) of the ground actions (see eq. 5.4). This heuristic is not admissible, i.e. is not a lower bound of the real cost. A similar heuristic h_c that is admissible can be obtained by estimating the maximum cost of achieving all the resources in x_0 (see eq. 5.5). The heuristics h_b and h_c are generally known respectively as Δ_0 and Δ_1 in the literature (Ghallab et al., 2004).

$$h_a(x_q) = |x_0| - |x_q \cap x_0| \tag{5.3}$$

$$h_b(x_q) = \sum_{p \in x_0} h_b(x_q, p) \tag{5.4}$$

$$h_b(x_q, p) = \begin{cases} 0, \text{ if } p \in (x_0 \vee x_q) \\ \min_{g_q^{a_h}}\{1 + h_b(x_q, \texttt{effect}^+(g_q^{a_h}) \mid p \in \texttt{condition}(g_q^{a_h})\} \end{cases}$$

$$h_c(x_q) = \max\{h_c(x_q, p) \mid p \in x_0\} \tag{5.5}$$

$$h_c(x_q, p) = \begin{cases} 0, \text{ if } p \in (x_0 \vee x_q) \\ \min_{g_q^{a_h}}\{h_c(x_q, p), 1 + \max\{h_c(x_q, q) \mid q \in \texttt{effect}^+(g_q^{a_h})\}\} \end{cases}$$

The estimated payoff function $\hat{\sigma}(g_q^{a_h}, x_q)$ is the weighted balance between the exact payoff function $\sigma(g_q^{a_h}, x_q)$ of the partial mission plan solution and the heuristic function $h(x_q)$ (see eq. 5.6).

$$\hat{\sigma}(g_q^{a_h}, x_q) = \alpha \cdot \sigma(g_q^{a_h}, x_q) + (1 - \alpha) \cdot h(x_q) \tag{5.6}$$

where α is the balance factor ($\alpha = 0.5$ by default).

5.3.2.2 Service-based Mission Planning

By distributing this candidate selection process, this approach can be applied to mission plan generation in a service discovery framework.

Given a set of services B, it can be seen that the total set of actions of a system A_V is the union of all the functionalities provided by all the services in the system (see eq. 5.7).

$$A_V = \bigcup_{b \in B} A_b \tag{5.7}$$

In this case, the set of candidate nodes W from which the planning process chooses the next state is the union of the actions that all the services can possibly provide from the current state (see eq. 5.8).

$$\begin{aligned} W &= \bigcup_{b \in B} W_b \\ W_b &= \bigcup_{g_q^{a_h} \in \{executable^{-1}(x_q) \cap A_b\}} \{\varphi^{-1}(x_q, g_q^{a_h})\} \end{aligned} \tag{5.8}$$

It can be seen that if the services are distributed among different agents in the platform, W can still be calculated for each given state x_{q-1}. In this case, the autonomous goal-based mission planning process previously described can be computed without having an explicit knowledge of A_V, i.e. without prior knowledge of the capabilities of the system. Furthermore, capabilities can be discovered dynamically as services join or leave the framework and the mission plan can be computed accordingly.

5.4 Architecture

This section describes the architectural components required to apply the service-oriented mission planning approach described in the previous section to a real platform scenario.

5.4.1 Goal-based Mission Planning Architecture

A description of the three layer architecture (Gat, 1998) that we implemented for the original autonomous mission planning approach (Patrón et al., 2008c) is shown in Figure 5.2. This figure represents a detailed version of the architecture diagram described

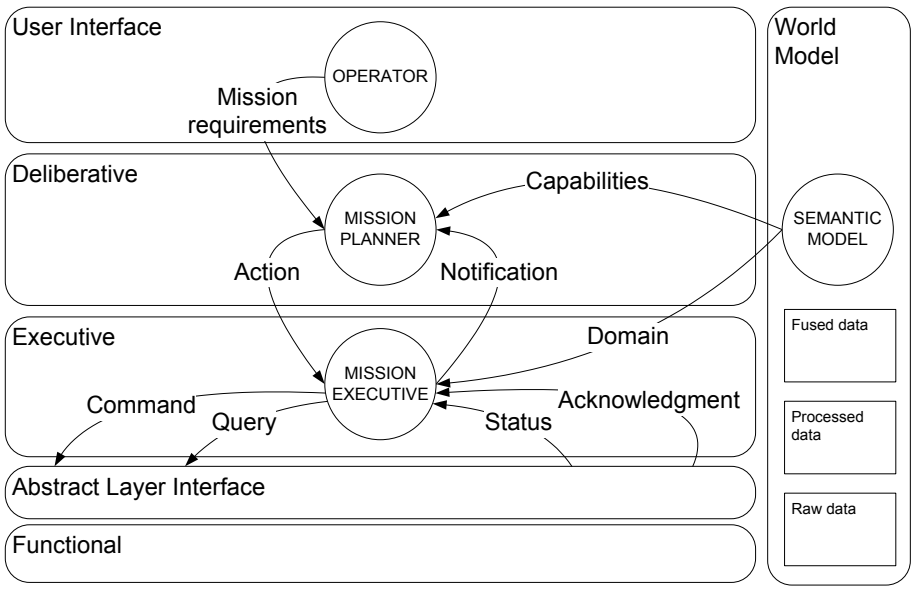


Figure 5.2: Original goal-based mission planning architecture: The semantic model provides the available platform capabilities to the mission planner agent, and the mechanism to execute the actions in the specific domain to the mission executive.

in Figure 1.6 for autonomous decision making. In this architecture, the world model stores the ontology-based knowledge containing the expert orientation and the observation data (see Section 2.2).

When the mission planner agent receives a new set of requirements from the operator, it generates a mission plan based on the known set of functionalities of the system reported by the Planning Application Ontology of the knowledge base. The sequence of ground actions of the calculated mission plan is then passed to the mission executive. This agent transforms the action in a set of commands according to the knowledge of the mission environment provided by the Core Ontology of the knowledge base.

The approach is detached from the custom properties of the functional layer of the host platform by an abstract layer interface (Tanenbaum, 1979). This makes the approach generic and platform independent.

5.4.2 Service-oriented Mission Planning Architecture

Figure 5.3 shows the approach proposed in this chapter for service-oriented mission planning. In this case, the service-oriented agents are assumed to be designed aware of their own capabilities and requirements. The following agents are defined:

- *Operator*: It includes the list of requirements for the mission from the operator.

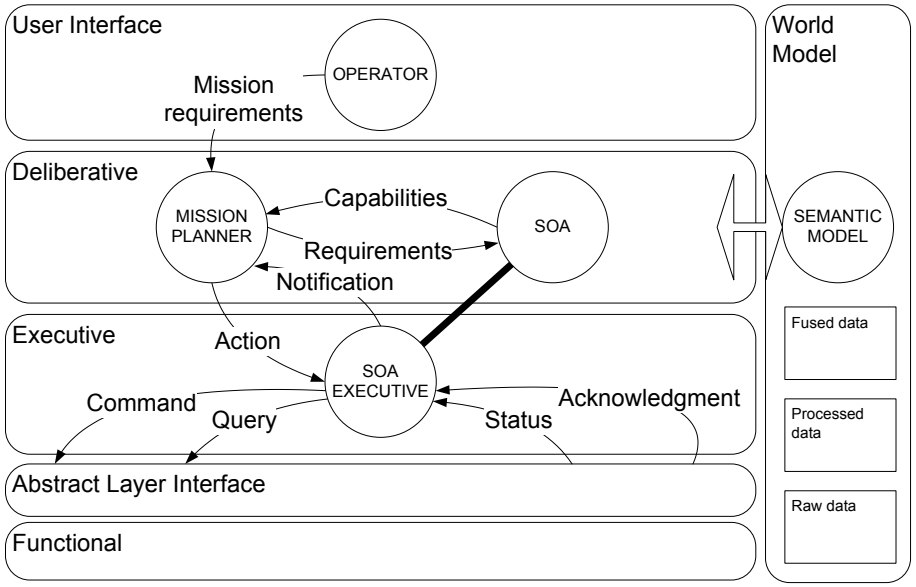


Figure 5.3: Proposed service-oriented mission planning architecture: Based on the requirement requests from the mission planner, the service oriented agents announce their capabilities via use of the semantic model representation. The mission plan execution is distributed by sending each instantiation of the actions to the corresponding service-oriented agent.

This agent can be linked to an operator interface in order to maintain the operator informed about the mission planning process and execution status (Johnson et al., 2007).

- *Planner*: This is the agent in the system which functionality is provided by the mission planning algorithms described in the previous section.

- *Service-oriented agents*: They are agents capable of communicating their capabilities and requirements. At the mission execution stage, they are also capable of performing their function upon request.

The mission planning process is calculated backwards in the deliberative layer from the mission requirements Q_O to the platform's current state x_0 using the service-oriented mission planning approach described in Section 5.3.2.2. In the executive layer, the mission execution process is performed forward following the sequence of instantiated ground actions of the calculated mission plan.

The discovery of capabilities is performed via coordination of the agents with the mission planner using a communication protocol whose messages use concepts defined

by the ontologies of the knowledge base. During the planning process, the following communication messages between agents have been identified:

- *Capability Request*: The mission planner agent issues a request when a new capability is required. This message can be seen as an instantiation of a requirement.
- *Capability Instantiation*: All SOAs that have a functionality capable of providing such capability reply to the request with one message for each possible instantiation of it and an associated estimated cost to it.
- *Requirement Acknowledge*: Each SOA informs to the mission planner that it has provided all the instantiations of its functionality that it has available for such request.
- *Planning Status*: The mission planner informs the other agents periodically of the status of the planning process.
- *Execution Status*: The mission planner informs the other agents periodically of the status of the execution process based on the notification messages coming from the SOA executives.

These last two status messages are not necessary for the correct performance of the approach but allow to keep the operator aware of progress.

For evaluation purposes, this messaging process has been implemented using the OceanSHELL UDP/IP broadcast protocol (OceanSHELL, 2005). These messages can be easily matched to some of the oncoming standards previously described in Section 5.3.1.

5.5 Evaluation of Service-oriented Mission Planning

This section describes how the service-oriented mission planning approach is able to solve a mission planning problem provided by an operator when the capabilities of the system are not known *a priori*.

The evaluation is based on the mine counter measure (MCM) operation scenario using UUVs. In this scenario, UUVs support and provide solutions for mine-hunting and neutralisation. The operation involves the process of Autonomous Target Recognition (ATR). This process is performed by the Computer Aided Detection (CAD) and

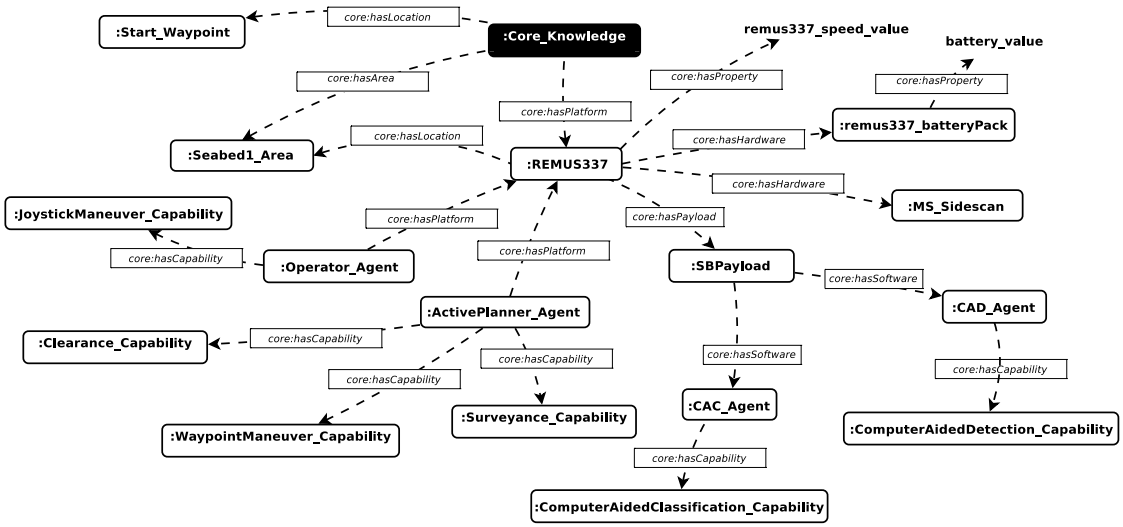


Figure 5.4: A subset of the instances in the Core Ontology describing the mission domain for the ATR scenario.

the Computer Aided Classification (CAC) agents, providing detection and classification functionality of mine-like objects respectively.

5.5.1 Knowledge Representation

Figure 5.4 shows a subset of the instances in the Core Ontology describing the mission environment for the ATR scenario. This figure represents the case where all agents and their capabilities are known *a priori* before starting the mission planning process. An equivalent situation awareness status is achieved in the discoverable environment after all services available in the platform have been published.

The representation of a subset of the planning concepts related to the mission plan and mission actions is shown in Figure 5.5. Note that this knowledge representation of planning concepts are linked to concepts already described in the Core Ontology, such as the list of capabilities required to perform each of the mission actions.

On the autonomous target recognition side, the two agents identified, CAD and CAC, make use of some of the elements of the Core Ontology such as: Sensor, Physical object, Seabed area, etc. The internal concepts of their correspondent Application Ontologies required to provide their detection and classification functionality are out of the scope of this chapter.

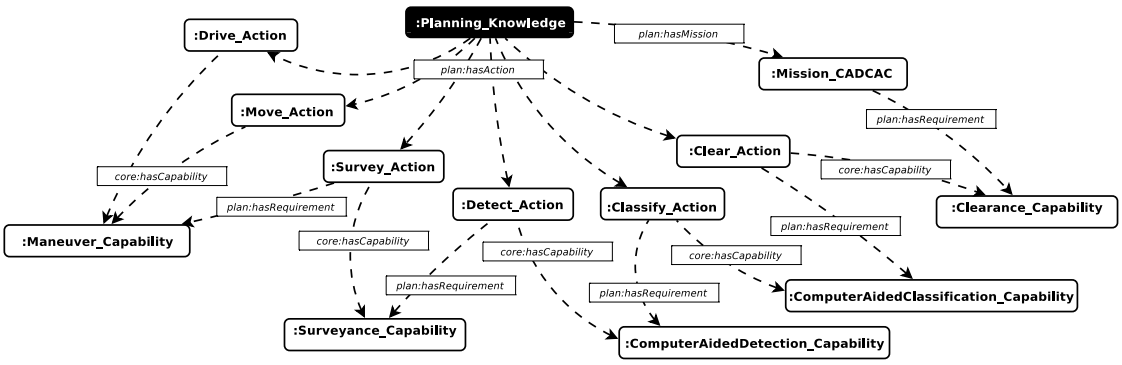


Figure 5.5: A subset of the instances describing the mission problem for the ATR scenario in the planner agent Application Ontology.

5.5.2 Illustration

This section illustrates how the proposed service-oriented mission planning approach operates under a discoverable service-oriented architecture. The decision making process occurs as a collaboration of the embedded service-oriented agents which, using the information available in the knowledge base framework, autonomously work out the chain of capabilities required to accomplish the mission requirements provided by the user.

Figure 5.6 shows an example of this process. In this figure, the arrows are oriented towards the direction of the backward planning process. The mission plan is defined by following the arrows in reverse order until the original operator requirements are achieved. By using the domain specific Core Ontology resident in the framework, all services can interchange information while remaining independent in their Application expertise.

In this simple example, the operator describes the mission goal for clearing an area of the seabed in a declarative manner and not in the classic procedural manner, i.e. 'what to' instead of 'how to' (step 1). This is passed to the adaptive mission planner agent that requires the classification of all possible mines in the area in order to perform such action (step 2). The computer aided classificator can provide such capability given the detection of mine-like objects in the area (step 3). The detection of objects requires a previous survey of the area with a sidescan sensor to be performed (step 4). The survey of the area can be provided by the functionality of two agents: either the mission planner executes a series of `toMove` actions that allow the vehicle to perform the survey (step 5.a) or the operator is tasked `toDrive` the vehicle in the area (step 5.b).

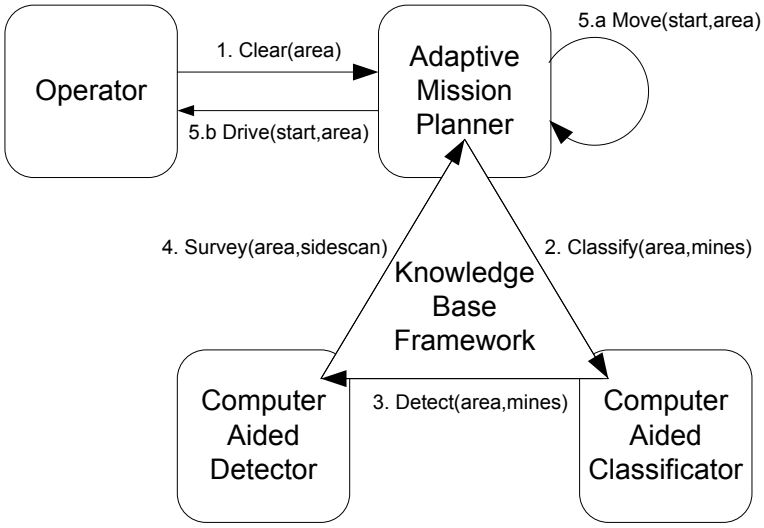


Figure 5.6: Interoperability between agents involving a chain of capabilities for achieving the mission requirements in a ATR scenario.

Assuming that the first option of this last step is chosen, the sequence of ground actions of the calculated mission plan is:

```
toMove (start, area)
toSurvey (area, sidescan)
toDetect (area, mines)
toClassify (area, mines)
```

The instances of the actions `toMove` and `toSurvey` will be executed by the mission planner agent. The instances of the actions `toDetect` and `toClassify` will be sent to the CAD and CAC agent respectively for execution.

5.5.3 Experimental Results

In order to test its performance, a set of 20 different synthetic ATR mission scenarios have been implemented. These scenarios target different metrics and mission goals. Their mission complexity is defined based on the number of elements for each of the mission properties. This complexity has been normalised and the mission problems sorted (see Figure 5.7). This type of representation aims to capture the level of mission complexity that forms part of autonomy level standards such as the one established by Huang et al. (2005).

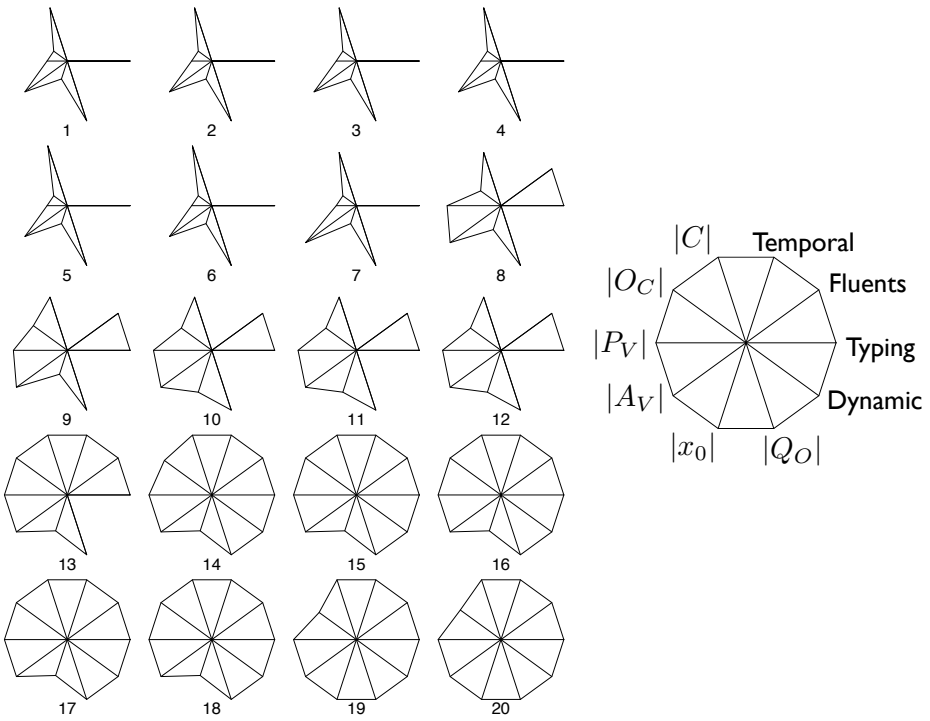


Figure 5.7: Graphical representation of the different ATR mission problems sorted by their level of complexity with legend describing each of the radius of the stars.

The legend from Figure 5.7 includes indication of the different properties of the mission environment. Starting from the top and in clockwise order, the Temporal, Fluents, Typing and Dynamic properties just capture the existence or not of action with duration properties, variables with numerical values, classification of objects and dynamism of the environment respectively. This last one indicates if there are capability changes during the planning process. Next, the number of mission requirements $|Q_O|$, the size of the current state $|x_0|$, the number of actions in the domain model $|A_V|$, the number of predicates $|P_V|$, the number of objects $|O_C|$, and the number of classes $|C|$ is represented. For these experiments, the maximum of these values has been limited to 30.

For evaluation purposes, each of the mission problems was represented using four files. Like in the previous chapter, this files are based on the PDDL syntax:

- the *Domain* file $\mathcal{D}$: describes the domain model Σ, including the classes in C, constants of O_C, the predicates P_V, the functions F_V and the actions A_V.
- The *Problem* file $\mathcal{P}$: describes the problem model Ω, including the initial state x_0, the rewards $\delta(O_C)$, the goals Q_O and the cost metric γ.

- The *Agent* file $\mathcal{A}$: describes the list of agents and which of the actions from $A_V \in \mathcal{D}$ are each of the agents capable to perform ($\mathcal{A} = \{B \cup (A_b | \forall b \in B)\}$.

- the *Environment* file $\mathcal{W}$: describes the correspondence between planning resources and the Core Ontology concepts and any possible attached procedure ($\mathcal{W} = \bigcup_{b \in B} core(b)$). This file is used during the execution only to translate some of the actions into commands that can be interpreted by the platform.

The reference and the service-oriented approach are computed for the 20 mission environments. We used a balanced factor of $\alpha = 0.5$ for the calculations of the estimated payoff function (see Eq. 5.6).

Both, reference and service-oriented approach find a mission plan for all the mission environments. Furthermore, both find the same mission plan. Figures 5.8 and 5.10 show the metric values of the final solutions for the reference and the SOA approach respectively. The value of these metrics is not relevant and it varies depending the metric considered, e.g. `distance` or `energy`. The relevant thing is that both approaches using the different heuristics present the same results to all the problems.

Figures 5.9 and 5.11 show the performance values for the reference and the SOA approach respectively. The main significant difference is that the computation time of the SOA approach is higher than the original approach. This is due to the communication process between agents negotiating the different sets of requirements. This can be observed by looking at the system activity represented in Figures 5.12 and 5.13.

Figures 5.12 and 5.13 show the system performance for the sequential computation of the mission planning process for the 20 scenarios using the original and the service-oriented approach. Looking at the time scale of these figures, it can be seen that the SOA approach takes longer than the original (approx. 6 times) due to the transmission over the network of an average of 5 packets/s. This produces a better distribution of the computational requirements, making the process less intensive. Additionally, the approach demonstrates that it follows the original requirement of not needing *a priori* knowledge of the system capabilities. As a consequence, the fact that it can take slightly longer to compute a solution is dismissed by the time savings provided by the removal of the manual reconfiguration process.

5.6 Summary and Outlook

Service-oriented architectures are being adopted in underwater robotics to allow integration of loosely coupled distributed capabilities. In order to provide mission flexibility and to maintain the platform's operability, it is important that the operator is released from the tedious task of having to communicate these changes on capabilities to the other intelligent agents. In this chapter we analysed how to exploit the dynamic discovery of system capabilities to autonomously generate mission plans under a service-oriented architecture.

First, we identified technologies and protocols allowing the dynamic publication of these capabilities. Secondly, we identified that the total set of actions of a system is the union of all the functionalities provided by all the services in the system. On this basis, we applied the semantic-based framework to a goal-based approach capable of providing autonomous mission planning from the dynamic discovery of the services published by the different agents running in the system. Our goal-based approach is based on backward state-space search from the mission objectives described by the operator to the current state of the mission environment. By distributing the selection of state candidates during the search process, this approach can be applied to autonomous mission planning in a service discovery framework when the capabilities of the platform are not known a-priori.

The approach was evaluated under a service discoverable scenario using a series of communication messages to publish the capabilities of the available agents. Results showed that this discovery-based implementation finds the same results as the baseline which was explicitly provided with the platform configuration. To the best of our knowledge, this is the first time that autonomous mission planning is used in combination with automatic service discovery to generate mission plans that cover the mission requirements and that agree with the capabilities discovered in the platform. This combination allows the software to be both platform independent, easing the manual creation of mission plans, and to be robust to dynamic changes in the platform configuration. These results provide an impact on mission flexibility, robustness and autonomy for UUVs.

This approach has also the potential to automatically provide further useful information to the operator. For instance, understanding the services that are not used by a mission plan can provide information about what payloads are redundant for a mission. Furthermore, identifying the services missing during the planning process can

indicate what additional payloads might be needed for a mission. This extension will be investigated in future work.

5.7 Key Publications

- The initial findings for distributed mission planning were presented in the paper *Cooperative planning architectures for multi-vehicle autonomous operations* during the **Conference for Systems Engineering for Autonomous Systems of the Defence Technology Centre** (SEAS-DTC'06) that took place in Edinburgh (Scotland) on July 2006 (Evans et al., 2006).

- The results from this chapter were described in the paper *Interoperability of agent capabilities for autonomous knowledge acquisition and decision making in unmanned platforms* during the **International Conference IEEE Oceans 2009 Europe** in Bremen (Germany) on May 2009 (Patrón et al., 2009b).

- The architecture behind the service-oriented mission planning approach was included in the paper *Multiple System Collaborative Planning and Sensing for Autonomous Platforms with Shared and Distributed Situational Awareness* appearing in the **Proceedings of the AUVSIs Unmanned Systems Europe** at the National Underwater Research Centre (NURC-NATO) in La Spezia (Italy) on June 2009 (Petillot et al., 2009).

- The evaluation of this architecture was presented in the paper *Situation-aware mission planning using distributed service oriented agents in autonomous underwater vehicles* at the **International Symposium on Unmanned Untethered Submersible Technology** in Durham, NH (USA) on September 2009 (Patrón et al., 2009c)

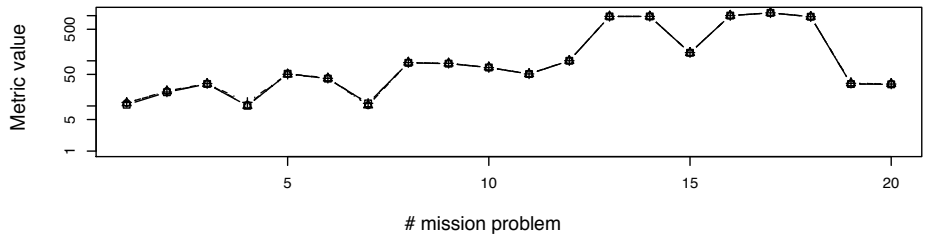


Figure 5.8: Final values of the domain specific metrics for the reference approach.

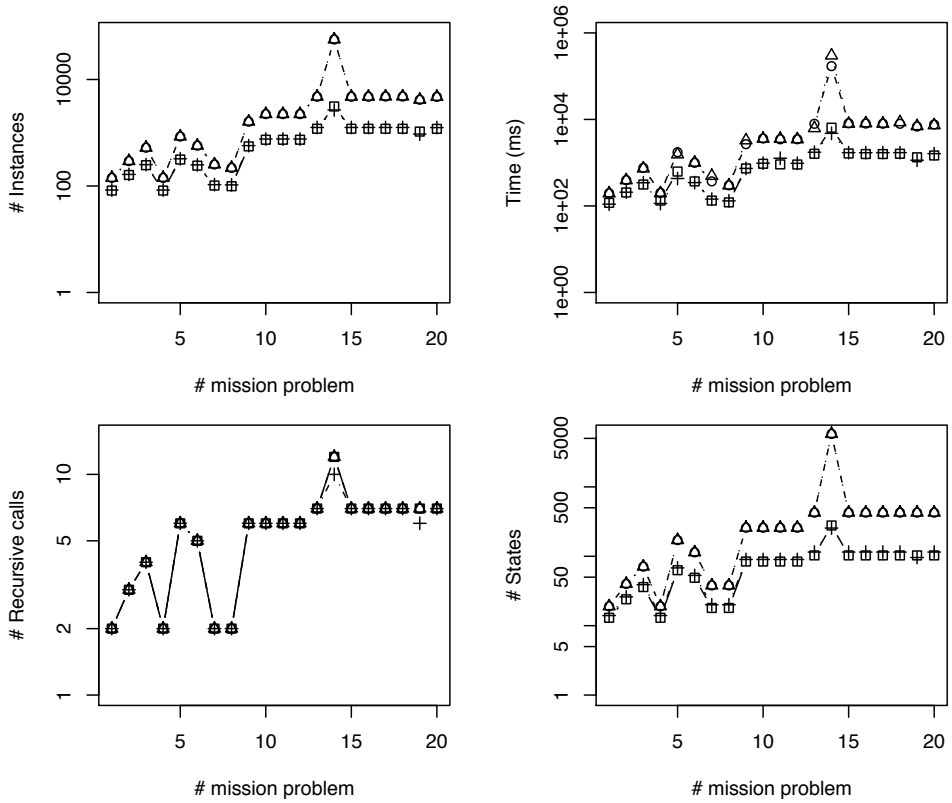


Figure 5.9: Search performance metrics (from top left to bottom right): Number of instances involved in the search process, computation time (ms), number of recursive calls and number of states calculated during the search process using $h_0 = 0$ ($\triangle$), h_a ($\circ$), h_b ($+$) and h_c ($\square$).

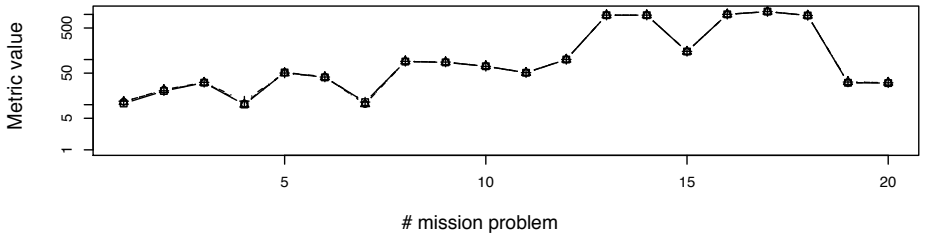


Figure 5.10: Final values of the domain specific metrics for the Service-oriented approach. It can be seen that for all the problems it obtains the same values as the reference approach results displayed in Figure 5.8.

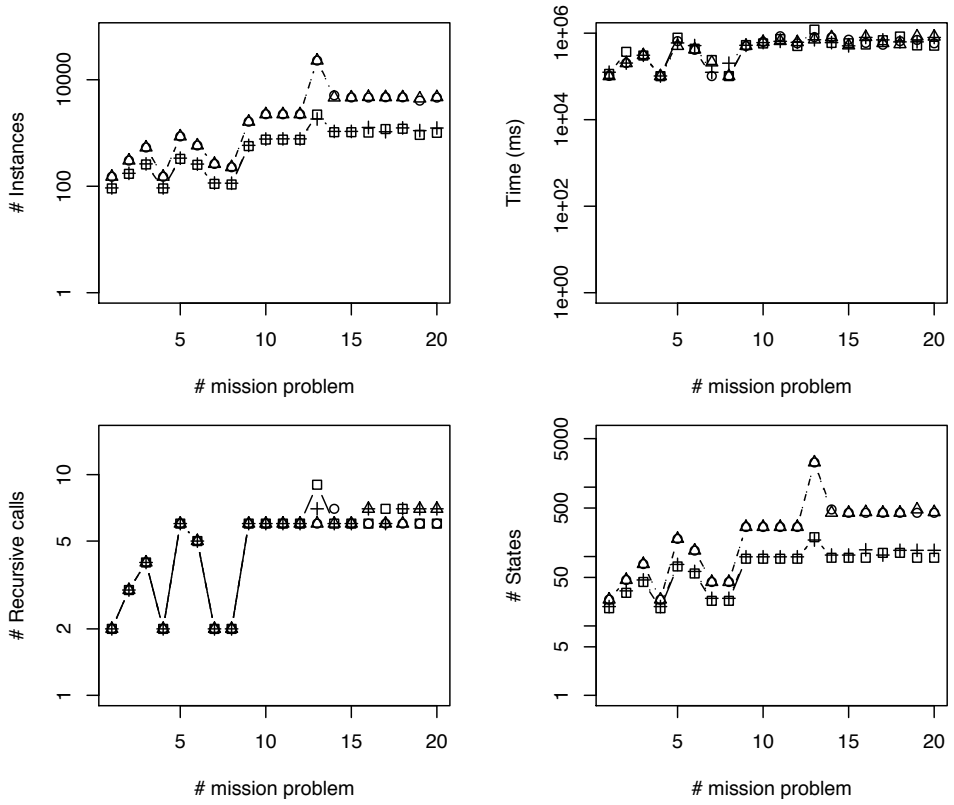


Figure 5.11: SOA search performance metrics (from top left to bottom right): Number of instances involved in the search process, computation time (ms), number of recursive calls and number of states calculated during the search process using $h_0 = 0$ ($\triangle$), h_a ($\circ$), h_b ($+$) and h_c ($\square$).

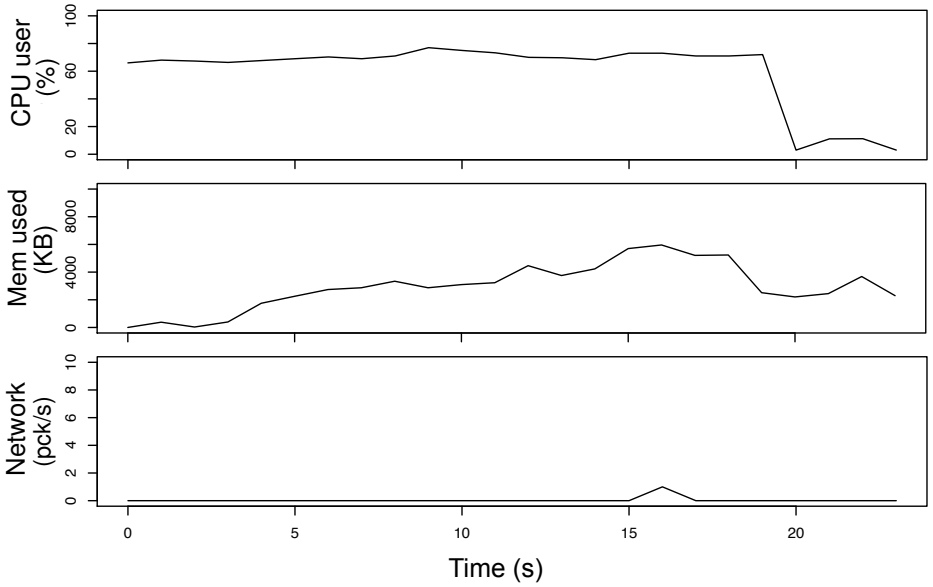


Figure 5.12: System activity during the execution of the reference approach. From top to bottom: % processor usage, memory usage (KB) and network activity (packets/s) over time (s).

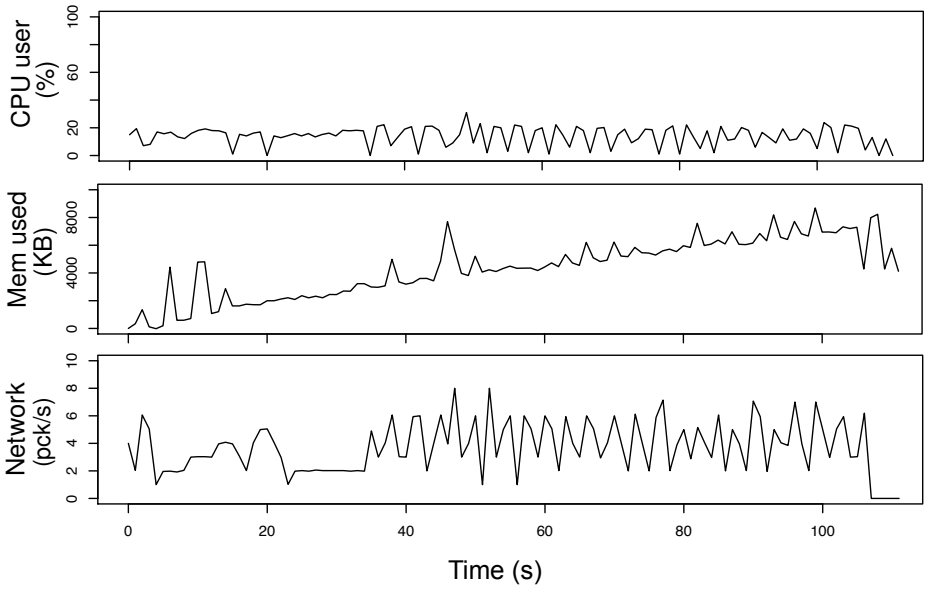


Figure 5.13: System activity during the execution of the service-oriented approach. From top to bottom: % processor usage, memory usage (KB) and network activity (packets/s) over time (s).

Chapter 6

Mission Plan Repair

Experience is a hard teacher because she gives the test first, the lesson afterwards.

– Vernon Sanders Law

6.1 Introduction

Generating a plan is a time consuming task. Normally, changes to the environment would require a regeneration of the entire plan. This can take too long to be able to handle constantly changing conditions. Also, the previous effort spent planning is wasted.

Under such circumstances, plan modification is an attractive solution as it can overcome these issues. The problem is that plan modification could be harder than plan regeneration, and it is not always possible to achieve an efficiency gain of reuse over generation (Nebel and Koehler, 1995). However, conservative approaches in the modification of the plan can minimise the perturbation of the original plan. This is very appealing for our scenarios, as it allows for the maintenance of commitments, and could increase human acceptance of autonomous planning in underwater platforms.

In previous work, we have demonstrated how re-using previous planning effort can provide efficiency and better results than re-planning from scratch. We applied this philosophy to produce minimal modifications of trajectory plans in UUVs by the combined use of lifelong planning and wave front propagation techniques (Patrón et al., 2005, 2007b).

In this chapter we study ways to extend this approach to goal-based declarative mission planning for autonomous decision making in UUVs. We focus on mission

plan adaptability methods for maximising response time under limited resources. We identify two ways of doing this: by re-using previous planning effort using mission plan repair techniques, and by allowing this mission repair to occur at the executive level when possible.

First, we implement a loop of unrefinement and refinement stages over a partial plan representation of the mission plan. By doing this, we achieve similar results in efficiency and adaptation for mission planning as we did in the past for trajectory planning. An added advantage of making minimal changes to a global plan as the mission progresses, is that the human operator knows that the vehicle will deviate only the minimum amount from a known plan. This could also be useful for a fleet of vehicles, where the actions of other vehicles need to be as predictable as possible. Mission plan repair allows for the maintenance of commitments, and increases human acceptance of autonomous planning.

Then, we show how to balance repair between the planning and the executive levels and how the semantic information can be critical in the decisions taken. Being able to cope with changes at the execution level saves the planner from computation tasks under limited computation resources. By combining these two techniques, our approach is capable of handling mission plan adaptation at the planning and execution levels maximising robustness, system performance and response time.

Finally, we implement a system that combines the benefits of knowledge-based ontology representation, autonomous mission plan repair and mission execution repair. The system combines plan adaptability and execution adaptability capabilities in order to provide robustness during the execution of a mission plan. Its performance is first presented in a set of simulated scenarios which show the benefits of mission plan repair vs. mission regeneration. Then the system is deployed as part of a payload in a real UUV providing adaptive seabed survey capabilities in a real test scenario. We report a set of sea trials where the robustness and autonomy of the approach is demonstrated. To the best of our knowledge, this is the first time that an approach to goal-based planning is applied to the adaptation of an underwater mission in order to maintain a platform's operability. Thus a major contribution of this thesis is demonstrating an integrated solution for adaptive mission planning in a real AUV.

6.2 Related Work

Several approaches can be found in the literature for solving the mission plan adaptability problem in a real scenario, where the importance of re-using previous effort and balancing deliberative and reactive responses to events has been implemented. As discussed in previous chapters, adaptive mission planning research has generally been motivated by Space exploration. We have found no evidence of mission plan repair being carried out for the underwater domain.

A simple approach to the re-use of previous effort is case-based planning. Case-based planning approaches have a case library of solutions to previous problems and uses these cases to solve new planning problems (Carbonell, 1984, 1993). Examples of this type of systems are PRIAR (Kambhampati and Hendler, 1992), SPA (Hanks and Weld, 1994), PRODIGY/Analogy (Veloso and Carbonell, 1993), and PARIS (Bergmann and Wilke, 1995).

Another approach is the use of plan repair techniques. As an example, the *CLEaR* (Fisher et al., 2000) described in Section 4.2 was later used by Chouinard et al. (2003) and Estlin et al. (2005) to highlight the difference between 'local' conflicts – errors that require changes only to the currently executing part of the plan – and 'global' conflicts – errors that occur which require changes to future parts of the plan. Local conflicts were managed by the executive while global ones were passed back to the planner to be fixed.

van der Krogt (2005) later formalised these two levels of handling events in what was called *executive repair* and *planning repair*, for the railway industry. The approach combined unrefinement and refinement stages over a partial plan representation. This provided faster performance than planning from scratch. However, this approach might fail to produce an optimal plan, which could be considered an issue in domains requiring optimality. This is not generally the case in unmanned vehicle mission plans where optimality can be sacrificed for operability. It is for this reason, that we adopt both the two levels of handling events, and the unrefinement and refinement approach to mission plan repair for UUVs.

van der Krogt's approach was compared with the *GPG* and the *Sherpa* systems. In the system *GPG* (Gerevini and Serina, 2000b), based on the *graphplan* planner (Blum and Furst, 1997), the unrefinement stage is done only on the initial plan and never on any of the plans produced by a refinement step. The *Sherpa* (Koenig et al., 2004), based on the LPA^* algorithm that we have previously used for adaptive trajectory plan-

ning (Pêtrès and Patrón, 2005; Patrón et al., 2007b) (see Section 1.2.3.4), could only be applied to problems in which actions have been removed from the model of the mission environment.

Recently, Fox et al. (2006b) proposed an on-board planning assistant to the operator to adjust and repair plans on-board. The system was developed to handle idle times due to conservative mission planning and plan failures on the unfortunate Beagle 2 Mars spacecraft. The approach relaxes methodological constraints and makes suggestions to the user on how to fill opportunity gaps, but only in situations where resources would otherwise go unused. A wider discussion on other related research that goes from strong executors and formal planning approaches to strong deliberators can be found in Knight's review of the field (Knight et al., 2001).

6.3 Mission Environment for Repair

This section describes the mission environment model used by the mission plan repair techniques presented in this chapter. We continue using the mission environment model described in the previous chapters. We generalise this model to any step q in the mission execution timeline. An instance of an UUV mission environment at a given step q can be simply defined as $\Pi_q = \{\Sigma_q, \Omega_q\}$. The mission domain model Σ_q contains the set of propositions defining the available resources in the system P_V and the set of actions or capabilities A_V. The mission problem model Ω_q contains the current platform state x_q and the mission requirements Q_O.

Based on this model, previous chapters have searched for mission plans on the state space. In this space the nodes were states of the mission environment and the actions were the arcs. In this chapter we analyse how to calculate mission plans on the plan space (Sacerdoti, 1975). A plan space is an implicit directed graph whose vertices are partially specified plans and whose edges correspond to refinement operations. In a real environment where optimality can be sacrificed by operability, partial plans are seen as a suitable representation because they are a flexible constrained-based structure capable of being adapted.

Definition 6.3.1 *A partial plan ψ is a tuple containing a set of partially instantiated actions and a set of constraints over these partially grounded actions. Constraints can be of the form of ordering constraints, interval preservation constraints, point truth constraints and binding constraints.*

Ordering constraints indicate the ordering in which the actions should be executed. Interval preservation constraints link preconditions and effects over actions already ordered. Point truth constraints assure the existence of precondition facts at certain points of the plan. Binding constraints on the variables of actions are used to ground the actions to variables of the domain. Figure 6.1 shows a partial plan representation of an UUV mission. It represents a scenario very similar to the one described in the demonstration in Section 6.7.2. A Remus UUV must navigate to two seabed areas, perform a survey with a sidescan sensor on the first area and reacquire the objects in the second area with a camera sensor.

While the state space is finite, the plan space is not. Also, unlike the state space, the intermediate states of the plan space do not provide an explicit representation of the state of the system. As a consequence, plan space search strategies do not return a plan but a subset of plans. However, partial plans are more flexible to modification. They provide an open approach for handling extensions such as temporal and resource constraints. Due to nature of the constraints, it is easier to explain a partial plan to a user than a sequence of actions in the state space. Another benefit is that it is easily extensible to distributed and multi-agent mission planning.

6.4 Mission Plan Adaptation

In this section we compare replan and repair techniques for mission plan adaptation using partial plan representations. We also discuss the benefits of repairing the mission at different levels. We define the mission plan adaptation process as follows. Given an instance of the mission environment Π_{q-1}, a solution mission plan for it π_{q-1}, and a variant of the mission environment for the same domain space Π_q ($\Pi_{q-1}, \Pi_q \in \Theta$), the goal of mission plan adaptation strategies consists in finding if there exists a mission plan π_q of length n formed by a sequence of ground actions $g_i^{a_h} \in \mathcal{G}_O, i \in [1,..,n]$, such that satisfies Π_q. The dynamic planning problem was first defined in Section 3.2.1. We can solve this problem either by generating a new mission plan after only considering the new mission environment, or by reusing the previous plan to cope with the new conditions. In previous chapters, we have only considered the changing mission environment. In the following sections, we present an approach to reusing the existing plan that exploits the flexibility provided by partial plan representation.

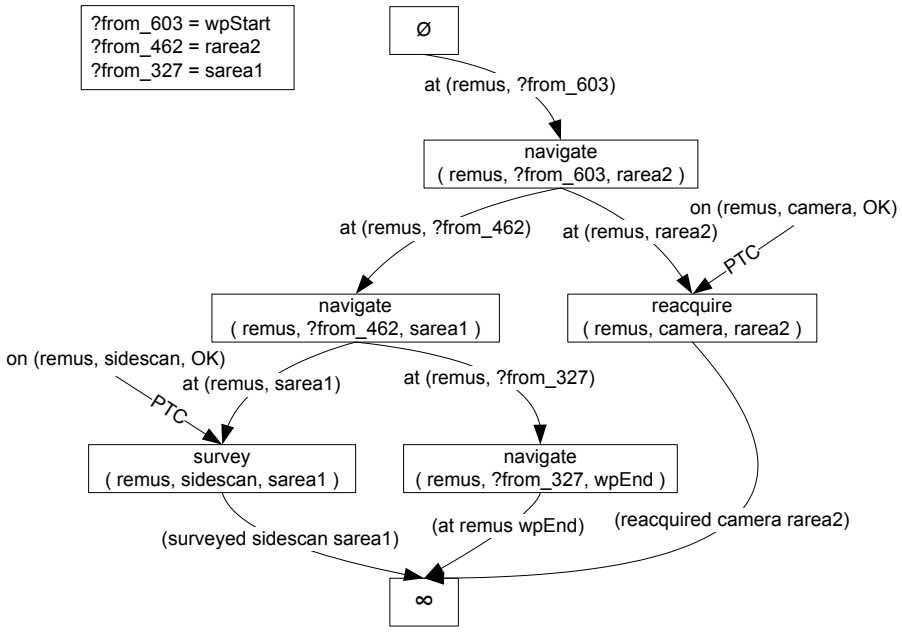


Figure 6.1: Example of a partial ordered plan representation of an autonomously generated UUV mission. The ordering constraints are represented using the graph depth, interval preservation constraints are represented with black arrows, point truth constraints are represented with PTC-labelled arrows, and binding constraints are shown in the top left box.

6.4.1 Replan vs. Repair

When a problem in a mission plan is detected, the aim is to be effective and efficient. A mission plan costs time and resources to prepare. This time has been already invested once (to compute the mission that is now failing), so it might be more efficient to try to reuse previous efforts by repairing the mission. Also, commitments might have been made to the current mission plan: trajectory reported to other intelligent agents, assignment of resources or assignment of part of mission plan to executors, etc. Repairing an existing mission ensures that as few commitments as possible are invalidated. Finally, several planners (usually autonomous and human planners combined) could be performing together to achieve the goals. In such cases, it is more likely that a similar mission plan will be accepted by the operator rather than a new one, that could be potentially completely different.

Figure 6.2 explains the processes of mission plan repair and mission plan replan for mission plan adaptation for UUVs using a partial plan representation of the mission

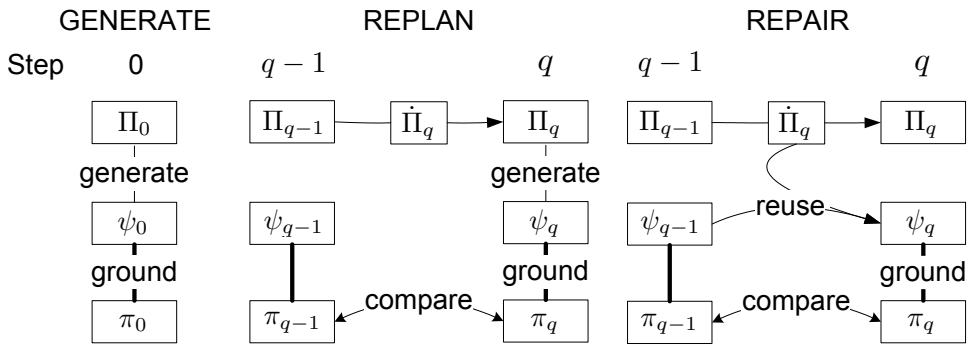


Figure 6.2: Schematic representation of the autonomous mission generation, replan and repair processes using partial plan representation of the mission plans.

plans. At the initial step, a partial ordered plan ψ_0 is generated satisfying the original mission environment Π_0. The ψ_0 is then grounded into the minimal mission plan π_0 including all constraints in ψ_0. At step q, the semantic knowledge-based framework is updated by the diagnosis information $\dot{\Pi}_q$ providing a modified awareness of the mission environment Π_q. From here, two mission adaptation processes are possible: *Mission replan* generates a new partial plan ψ_q, as done at the first stage, based only on the knowledge of Π_q. On the other hand, *mission plan repair* re-validates the original plan by ensuring minimal perturbation of it. Given the partial plan at the previous step ψ_{q-1} and the diagnosis information $\dot{\Pi}_q$, the mission repair problem produces a solution partial plan ψ_q that satisfies the updated mission problem Π_q, by modifying ψ_{q-1}. The final step for both approaches is to ground ψ_q to its minimal mission plan π_q. It can be seen that mission repair better exploits the orientation capabilities for decision making: instead of taking the new mission environment as a given, it uses the diagnosis information about the changes occurred to guide the adaptation process.

6.4.2 Plan repair vs. Executive repair

We have now identified the benefits of mission plan repair over mission replan. Mission plan repair modifies the partial plan ψ_q, so that it uses a different composition, though it still maintains some of the actions and the constraints between actions from the previous partial plan. However, mission plan adaptation can also be achieved by mission execution repair by looking directly at the mission plan instantiation π_q. Execution repair modifies the instantiation of the mission plan π_q such that a ground action $g_q^{a_h}$ that was previously instantiated by some execution e_q is newly bound by another

action execution instance e'_q.

Executive repair is less expensive and it is expected to be handled directly by the mission executive agent. Plan repair, however, is computationally more expensive and requires action of the mission planner agent.

The objective is to maximise the number of execution repairs over plan repairs and, at the plan repair level, maximise the number of decisions reused from the previous mission instantiation. The information provided by the semantic-base knowledge base during the plan diagnosis phase is critical.

6.5 Mission Plan Repair

In the previous section we identify the alternatives to mission plan adaptation and their different properties. This section focuses on our approach to mission plan repair for unmanned underwater vehicles. Mission plan repair involves the detection of events, the diagnosis of the effects that these events have on the mission plan and the response phase. As discussed, we based our approach on plan recovery methods (van der Krogt, 2005) and extended them to the underwater domain. Plan recovery methods are based on plan-space searches and are able to adapt the existing plan to the new state of the world. We have extended these methods by combining them with the semantic knowledge-based framework introduced in Chapter 2. This combination provides a system that is suitable for the type of modular and dynamic solutions required in the underwater environment. Our research also goes beyond van der Krogt's approach by providing an evaluation of the combined solution under a practical realistic application. In the following sections we describe the implementation of our approach for the three stages of detection, diagnosis and repair.

6.5.1 Detection

For the detection phase, we use the functionality of a status monitor agent in combination with the knowledge-base framework described in Chapter 2. As we presented in that chapter, the status monitor agent considers all symptoms and observations from environmental and internal data in order to identify and classify events according to their priority and their nature (critical or incipient). Based on internal events and context information, this agent is able to infer new knowledge about the current condition of the vehicle with regard to the availability for operation of its components, i.e. sta-

tus (Hamilton, 2002). In a similar way, environmental data is also considered for detecting and classifying external events in order to keep the situation awareness stored in the knowledge base updated.

6.5.2 Diagnosis

Based on the analysis of the effects of the updated mission environment on the current mission plan, the plan diagnosis phase identifies the failures and gaps existent in the current mission plan. Plan failures are unsuccessful executions of actions. Plan gaps are parts of the plan that are no longer executable. Identifying these two elements makes repairing the plan possible, as they are the cause of the inconsistencies between the existent plan and the current status of mission environment. They are, therefore, preventing the correct execution of the mission. Our approach uses the reasoning and querying capabilities of the semantic knowledge-based framework presented in Chapter 2 to identify the impact that changes in the mission environment reported by the status monitor have on the mission plan. A detailed explanation on how the outputs from the status monitor are used to identify the level of repair and the gaps in the mission plan can be found in the description of the field experiment presented in Section 6.7.2.

6.5.3 Executive Repair

Executive repair fixes plan failures identified in the mission plan during the plan diagnosis stage. Our approach uses ontology reasoning in combination with an action execution template to adapt the mission plan at the executive level.

Once a mission plan π_q is calculated by the mission planner, its list of ground actions is transferred to the executive layer. In this layer, each ground action $g_q^{a_h}$ of π_q gets instantiated into an action execution instance e_q^t using the action template for the action a_h available in the Core Ontology of the knowledge base. At the end of this phase, each e_q^t contains the script of commands required to perform its correspondent ground action (see Figure 6.3). Flexibility in the execution of an action instance is critical in real environments. This is provided by a timer, an execution counter, a time-out register and a register of the maximum number of executions in the action execution instance. Additionally, three different outputs control the success, failure or time-out of its execution. These elements handle the uncertainty during the execution phase and enable the executive repair process. This minimise the number of calls to

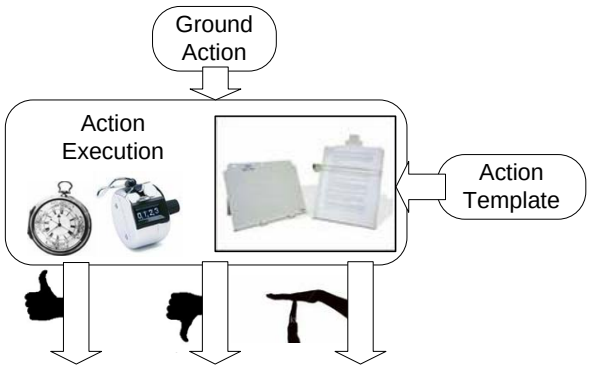


Figure 6.3: An action execution e_q^t is instantiated from a ground action $g_q^{a_h}$ from the mission plan using the action a_h template from the Core Ontology of the knowledge base. Each instance contains a list of commands, a timer, an execution counter, a time-out register and a register of the maximum number of executions. The success, failure and time-out outputs point at the next instance to be executed in the mission plan.

the adaptive mission planner agent and therefore the response time for adaptation.

6.5.4 Plan Repair

Plan repair uses a strategy to repair with new partial plans the plan gaps identified during the plan diagnosis stage. Our approach uses an iteration of unrefinement and refinement strategies on a partial-ordered planning framework to adapt the mission plan at the planning level.

Planning in the plan space is slower than in the state space because the nodes are more complex. Refinement operations are intended to achieve an open goal from the list of mission requirements or to remove a possible inconsistency in the current partial plan. These techniques are based on the least commitment principle, and they avoid adding to the partial plan any constraint that is not strictly needed. A refinement operation consists of one or more of the following steps: adding an action, an ordering constraint, a variable binding constraint or a causal link.

A partial plan is considered to be a solution to the planning problem if has no flaw and if the sets of constraints are consistent. Flaws are either subgoals or threats. Subgoals are open preconditions of actions that have not been linked to the effects of previous actions. Threats are actions that could introduce inconsistencies with other actions or constraints. We implemented a recursive non-deterministic approach based on the Partial ordered Planning (PoP) framework (Penberthy and Weld, 1992). This

framework is sound, complete, and systematic. Unlike other Plan space planners that handle both types of flaws (goals and threats) similarly, each PoP recursive step first refines a subgoal and then the associated threats (Ghallab et al., 2004).

In our implementation, we introduce a previous step capable of performing an unrefinement of the partial plan when necessary. During the unrefinement strategy we remove refinements from the partial plan that are reported by the plan diagnosis phase to be affecting the consistency of the mission plan with the mission environment, i.e. to remove constraints and finally the actions if necessary.

Figure 6.4 shows a graphical representation of the mission plan repair process. In simple terms, when changes on the *ABox* Planning Application Ontology are sensed ($\dot{\Pi}_q$) that affect the consistency of the current partial plan ψ_{q-1}, the plan repair process is initiated. The plan repair stage starts an unrefinement process that relaxes the constraints in the partial plan ψ_{q-1} that are causing the mission plan to fail.

The remaining temporal mission partial plan ψ'_{q-1} is now relaxed to be able to cope with the new mission environment. However, this relaxation could open some subgoals and introduce threats in the partial plan that need to be addressed. The plan repair stage then executes a refinement process searching for a new mission plan ψ_q that is consistent with the new mission environment Π_q and removing these possible flaws. By doing this, it can be seen that the new mission plan ψ_q is not generated again from Π_q (re-planned) but recycled from ψ_{q-1} (repaired). This allows re-use of the parts of the plan ψ_{q-1} that were still consistent with Π_q.

6.6 Semantic-based Adaptive Mission Planning System

The combination of the status monitor agent, the adaptive mission planner, the mission executive and the semantic knowledge-based framework is termed as the Semantic-based Adaptive Mission Planning system (SAMP). The SAMP system implements the four stages of the OODA-loop described in Section 1.4. Figure 6.5 represents the customised version of Figure 1.6 for SAMP.

The status monitor agent reports to the knowledge base the different changes occurring in the environment and the modifications of the internal status of the platform. The knowledge base stores the ontology-based knowledge containing the expert orientation provided *a priori* and the observations reported by the status monitor. A mission planner agent generates and adapts mission plans based on the situation awareness stored in the knowledge base. The mission executive agent executes mission commands in

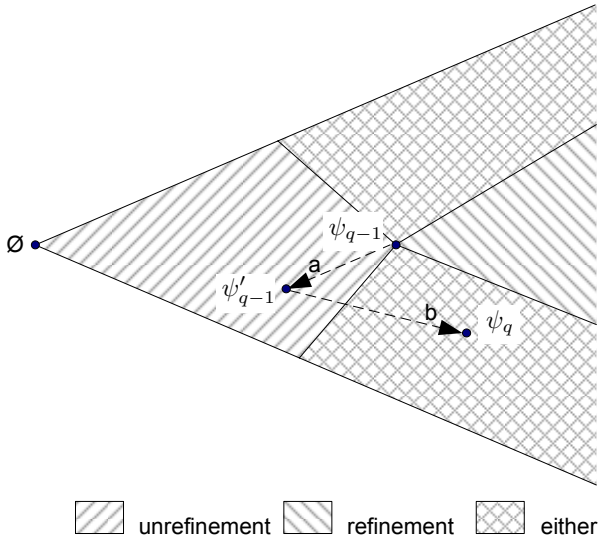


Figure 6.4: Representation of the refinement and unrefinement search process over the plan space domain. Starting from a partial plan ψ_q, the arrow line a shows the unrefinement process to achieve a relaxed and modified partial plan ψ'_{q-1}. From there, line b shows the refinement process to obtain the partial plan considering the new constraints ψ_q. The symbol $\emptyset$ shows the empty plan.

the functional layer based on the sequence of ground actions received from the mission planner. An Abstract Layer Interface (ALI) based on JAUS-like messages (SAE-AS-4 AIR5665, 2008) over UDP/IP packages implemented using the OceanSHELL protocol (OceanSHELL, 2005) provides independence from the platform's functional layer making the system generic and platform independent.

6.7 Evaluation of Mission Plan Recovery

This section evaluates the performance of the SAMP system for adaptive mission planning in a set of synthetic simulations and as part of a real application experiment.

6.7.1 Simulation

A set of synthetic simulated scenarios have been implemented in order to test the performance of the SAMP system. The tests are based on the mine counter measure (MCM) operation using AUVs described as part of the reactive data gathering scenario presented in Section 1.3.2.2. The operation involves high levels of uncertainty and risk of damage to the vehicle. Navigating in such a hazardous environment is likely

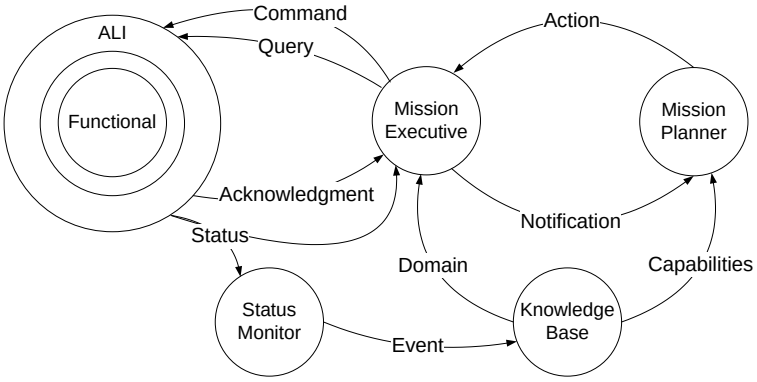


Figure 6.5: Architecture of the SAMP system. The embedded agents are the planner, executive, monitor, and knowledge base. These agents interconnect via set of messages. The system integrates to the functional layer of a generic host platform by an abstract layer interface (ALI).

to increase the vulnerability of the platform. Additionally, if a vehicle is damaged or some of its components fail, mission adaptation will be required to cope with the new restricted capabilities.

A set of 15 selected MCM scenarios were simulated. These scenarios were designed to cover the variability of missions described by the concepts of operations for unmanned underwater vehicles presented in the UUV Master Plan (2004) and the JRP Master Plan (2005). For each scenario, the detection of a failure in one of the components of the system was simulated. The mission plan was adapted to the new constraints using replanning methods and the mission plan repair approach based on partial plans introduced in Section 6.5.

The performance of the two approaches was compared by looking at the computation time and the Plan Proximity of the adaptive mission plan provided to the original reference mission plan. Figure 6.6 left shows the computation time in milliseconds required for adapting the mission to the new constraints for replan (dark grey bars) and repair approaches (light grey bars). Note that a logarithmic scale is used for the time values. Figure 6.6 right displays the Plan Proximity to the original plan of the replan strategy result versus the repair strategy result. It can be seen that plans adapted using the mission repair strategy tend to be closer to the original plan than using the mission replan strategy. In these results, 14 out of 15 scenarios were computed faster by using mission plan repair. This computation was on average $9.1x$ times faster. Also, 14 out of 15 scenarios showed that mission plan repair had greater or equal Plan Proximity val-

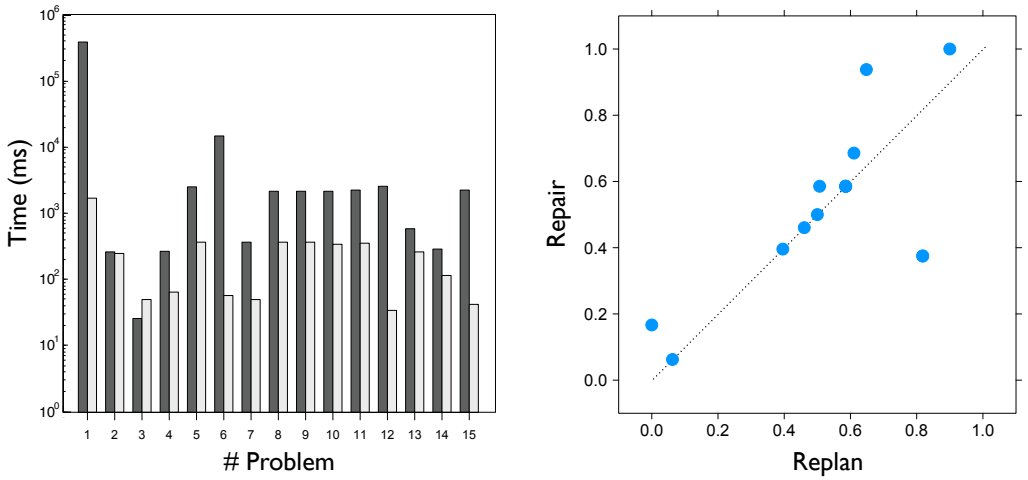


Figure 6.6: Left: A semi-log plot displaying the computational time in miliseconds for replan (dark grey bars) and repair approaches (light grey bars). Right: Comparison of Plan Proximity ($PP_{0.5}$) of the replan and repair approaches to the original plan.

ues as compared to mission replan. In general, our mission repair approach improves performance and time response while at the same time finds a solution that is closer to the original mission plan available before adaptation.

6.7.2 Field Experiments

The goal of this experiment is to demonstrate the capabilities of the SAMP mission plan repair system inside a real UUV. We show how the operator can benefit from using a semantic representation of the environment's situation and a high level goal-based description of the mission. The experiment also demonstrates the benefits of *in situ* mission adaptation using semantic-based knowledge representation and interactions between the status monitor agent and the adaptive mission planner agent.

The performance of the SAMP system was evaluated on a REMUS 100 AUV platform (see Figure 6.7) in a set of integrated in-water field trial demonstration days at Loch Earn, Scotland ($56^o23.1'$N,$4^o12.0'$W). The REMUS AUV had a resident guest PC/104 1.4GHz payload computer where the SAMP system was installed. SAMP was capable of communicating with the vehicle's control and status modules and taking control of it by using an interface module that translated the ALI protocol messages into the manufacturer's *Remote Control* protocol messages (REMUS AUV RECON v1.12, 2008).

Figure 6.7: REMUS 100 AUV deployment before starting one of its missions.

Figure 6.8 shows the procedural waypoint-based mission as it was described to the vehicle's control module. This was known as the baseline mission. It was only used to start the vehicle's control module with a mission in the area of operation before taking control of it using the SAMP system. The baseline mission plan contained a start waypoint and two waypoints describing a North to South mission leg at an approximate constant Longitude ($4^o16.2'$W). This leg was approximately 250 meters long and it was followed by a loiter pattern at the recovery location. The track obtained after executing this baseline mission using only the vehicle control module is shown in Figure 6.8 with a dark line. A small adjustment of the vehicle's location can be observed on the top trajectory after the aided navigation module corrects its solution to the fixes received from the Long Baseline (LBL) transponders previously deployed in the area of operations (see Section 1.2.3.4 for more information about UUV navigation solutions).

On the payload side, the SAMP system was oriented (in the OODA-loop sense) using *a priori* information about the environment, the platform and a declarative description of the goals of the mission. The *a priori* knowledge was represented using the Core Ontology described in Section 2.3.2.1. Knowledge about the platform configuration capabilities is displayed in Figure 2.11 in Chapter 2 using Core Ontology concepts. Knowledge about the environment was provided based on automatic computer-aided seabed classification information generated from previous existent data (Reed et al., 2006b). The two classified seabed areas are shown in Figure 6.8. The declarative description of the mission requirements was represented using concepts from the

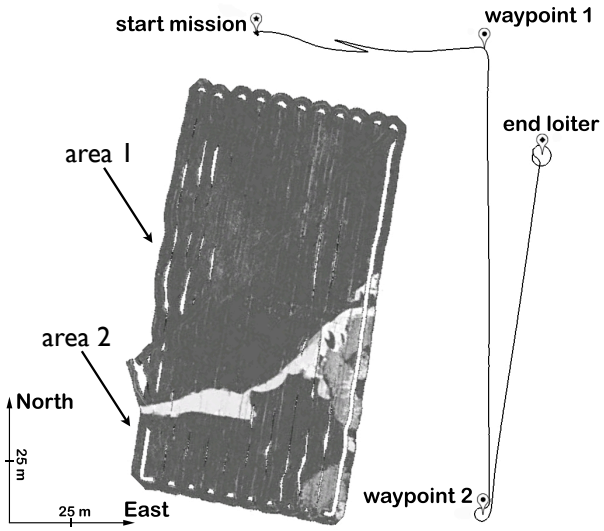


Figure 6.8: Procedural mission uploaded to the vehicle control module and *a priori* seabed classification information stored in the knowledge base. The two dark grey areas correspond to the classified seabed regions.

Planning Application Ontology described in Section 2.4.2. They could be resumed as 'survey all known areas maximising efficiency'. For experimental purposes, efficiency was considered as the combination of battery usage and distance travelled.

Following the reasoning process in the semantic-based knowledge framework described in Section 2.5.1, the SAMP system was capable of autonomously infer that the existent platform configuration was suited to attempt the generation of a mission plan that can fulfil the mission requirements provided by the operator. This is a very valuable assistance to operators that are not involved with the engineering process of the platform. Note that, although not integrated in this experiment, if the service discovery capabilities described in Chapter 5 were incorporated, the system would have been able to answer these pre-mission questions even when it was not fully oriented about the overall set of functionalities provided by all the payloads available in the platform. This approach provides a novel way of operating UUVs in which scientific operators can concentrate on the simple declarative description of the mission and focus on the post-processing of the data gathered. At the initial point the system was made unaware of the seabed classification information. As a consequence, the initial mission plan was generated using a refinement only approach and guiding the vehicle directly towards the recovery point.

For these experiments, SAMP was given a static location in which to take control of the host vehicle. At this point the new knowledge about the seabed areas was made

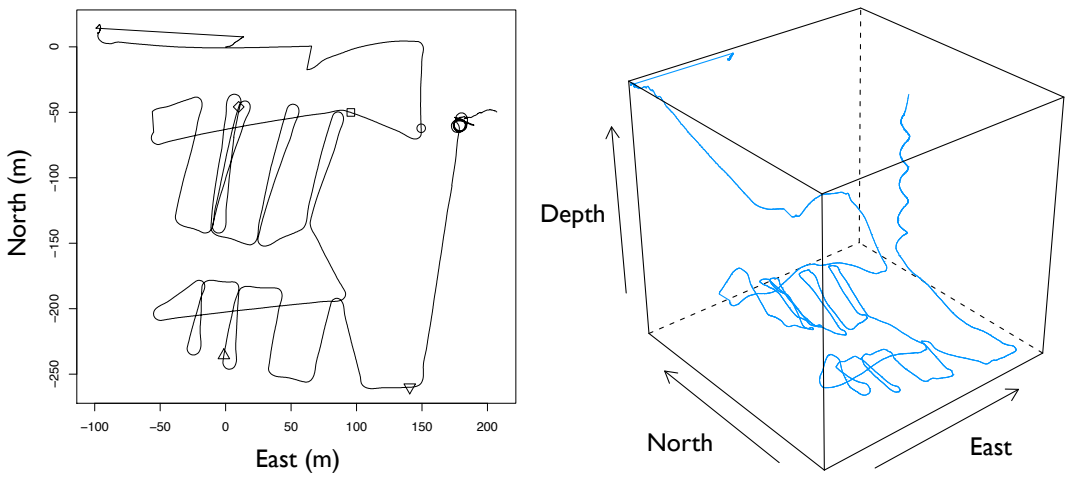


Figure 6.9: Left: Vehicle's track during mission in a North-East coordinate frame projection with the origin at the starting point of the mission. Right: Three-dimensional display of the vehicle's track during the mission (Note that depth coordinates are not to scale).

available. As a consequence, the mission planner agent repaired the mission plan to take this new knowledge into consideration. During this process, a small unrefinement step relaxed the plan by removing the constraints and action that was driving towards the recovery point. Then, a set of refinement iterations added the necessary actions and constraints that allowed to perform the survey of the two areas. The instantiation of this adapted mission plan is described in Figure 2.12 of Section 2.5 using Planning Application Ontology concepts. The mission was then passed to the executive agent that took control of the vehicle for its execution.

While the mission was executed the status monitor agent maintained the knowledge base updated with new observations (in the OODA-loop sense) by reporting changes in the status of hardware components, e.g. batteries and sensors, and external parameters, e.g. water currents.

When observations indicated that some of these changes were affecting the mission under execution, the mission planner was activated in order to adapt the mission to the changes. This indication was detected by the planner agent by querying the knowledge base using the queries described in Section 2.5.2.

The aim of this experiment was to show how the SAMP system was able to adapt the current mission plan to changes in the status of the internal hardware components and external parameters.

An internal fault was simulated by dropping the gains of the starboard transducer of the sidescan sonar below their minimum levels half way through the lawnmower survey

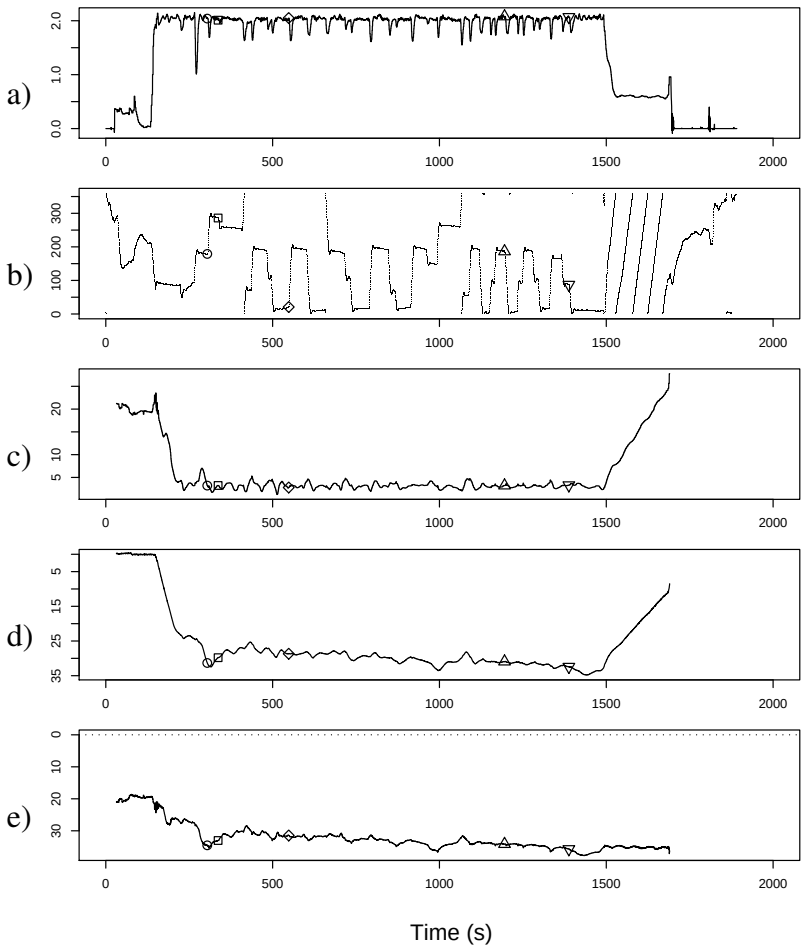


Figure 6.10: Vehicle telemetry (top to bottom): a) vehicle velocity (m/s), b) compass heading (degrees), c) altitude (m), d) depth (m) and e) reconstructed profile of the seabed bathymetry (m) during the mission, all plotted against mission time (s).

of the first area. Due to the fact that the sidescan sonar component is composed of two transducers, port and starboard, one malfunctioned transducer was only diagnosed as an incipient *Status* of the overall sidescan sonar component (see Section 2.5.2). An incipient *Status* of the action parameters indicates that the action can still be performed by adapting the way it is being executed, an execution repair. If both transducers were down, a critical status of the sidescan sensor is diagnosed and a plan repair adaptation of the mission plan would have been required instead. In that case, the adaptive mission planner would have looked for redundant components or similar capabilities to perform the action or to drop the action from the plan.

These gain levels were re-established back to normal values at a location situated in the middle of the second area. The same procedure was used after the transducer recovery was reported to adapt the survey action to the normal pattern during the second

lawn mower survey.

In a similar process, SAMP adapted the lawnmower pattern survey of the areas to the detected water current *Status* at the moment of initialising the survey of the areas.

The timeline of the mission executed using the SAMP approach is described in the following figures: Figure 6.9 represents the final trajectory of the vehicle in 2D and 3D using a North-East coordinate frame projection with the origin at the starting point of the mission. Figure 6.10 displays the vehicle's telemetry recorded during the mission. It includes vehicle's velocity, compass heading, altitude, depth measurements and processed bathymetry over time. Figure 6.11 shows subset of the variables being monitored by the status monitor agent that were relevant to this experiment. These variables include direction of water current, remaining battery power, the availability of the transducers in the sidescan sensor and the mission execution status. Figure 6.12 represents the system activity of the payload computer recorded during the mission. The system activity logs show percentage of processor usage, memory usage, network activity and disk usage.

Each of the symbols ◯, □, ♢, △ and ▽ on the aforementioned figures represents a point during the mission where an event occurred. Symbol ◯ represents the point where SAMP takes control of the vehicle. Note a change on the host platform mission status binary flag that becomes 0x05, i.e. the mission is active (0x01) and the payload is in control (0x04) (Figure 6.11.e). Also, a peak on the CPU usage can be noted as this is the point where the mission partial plan gets generated (Figure 6.12.a).

Symbol □ represents the point where the vehicle arrives to perform the survey of the area. At this point, the action survey gets instantiated based on the properties of the internal elements and external factors. Although the Loch waters where the trials were performed were very still (see Figure 6.11.b), note how the vehicle heading during the lawnmower pattern performed to survey the areas follows the water current direction sensed at the arrival (approx. 12^o, Symbol □ - Figure 6.11.a) in order to minimize drag and maximise battery efficiency. The heading of the vehicle during the survey can be observed in Figure 6.9 and Figure 6.10.b. The link between the vehicle heading in relation to the water current direction and its effect on the battery consumption was expert orientation knowledge captured by a relationship property between the two concepts in the Core Ontology.

Symbol ♢ represents the point when the status monitor agent detects and reports a critical status in the starboard transducer of the sidescan sonar (Figure 6.11.d). It can be seen how the lawnmower pattern was adapted to cope with the change and to use

the port transducer to cover the odd and even spacing of the survey. This pattern avoids gaps in the sidescan data under the degraded component configuration and maximises sensor coverage for the survey while the transducer is down.

Symbol $\triangle$ indicates the point where the starboard transducer recovery is diagnosed. It can be observed how the commands executing the action are modified in order to optimise the survey pattern and minimise distance travelled. Although also being monitored, the power status does not report any critical status during the mission that requires modification of the actions (Figure 6.11.c).

Symbol $\triangledown$ shows the location where all the mission goals are considered achieved and the control is given back to the mission control of the host vehicle (see Symbol $\triangledown$ - Figure 6.11.e shows the mission is still active but the payload is not longer in control (0x01)). From this point the host vehicle's control module takes the control back and drives the vehicle to the loiter at the recovery location.

6.8 Summary and Outlook

In this chapter we identified that regeneration of the mission plan when changes occur is not always possible. On this basis we compare replanning, or mission plan regeneration, and mission plan repair techniques. Mission plan repair is capable of modifying already available mission plans to cope with the new environmental conditions. We proposed a mission plan repair implementation based on a partial mission plan representation using an iteration of unrefinement and refinement techniques. Unrefinement is used to remove the constraints affecting the current plan under the new conditions. Refinement is then used to add the new constraints required to cope with the new conditions.

By using the Plan Proximity metric, we showed that repairing a mission plan tends to provide faster and closer solutions to the original mission plan than replanning. We also looked at minimising the plan repair by providing mission repair at the executive level. We achieve this by making use of alternative instantiations of the mission plan actions.

Finally, we developed and deployed an implementation of a semantic-based autonomous planning system in a real underwater vehicle. This system combines a semantic-based hierarchical representation of knowledge and mission plan adaptation techniques. The advantage of this combination is that it maximises robustness, system performance and response time. By making use of an abstraction layer interface, the

approach is platform independent making it readily applicable to other domains, such as ground or air vehicles.

This is the first time that an approach to goal-based planning is applied to the adaptation of an underwater mission in order to maintain platform's operability. The results showed adaptability to environmental elements, such as water current flows in order to improve mission performance. The approach was also capable of dealing with changes in status of certain components in the platform and was able to react accordingly.

6.9 Key Publications

- A principles for mission plan repair presented in this chapter were presented in the paper *Mission plan recovery for increasing vehicle autonomy* during the **Conference for Systems Engineering for Autonomous Systems of the Defence Technology Centre** (SEAS-DTC'07) that took place in Edinburgh (Scotland) on July 2007 (Patrón et al., 2007a).

- The elements of the semantic-base adaptive mission planning system were included in the paper *Adaptive mission planning: The embedded OODA loop* presented at the **Conference for Systems Engineering for Autonomous Systems Defence Technology Centre** (SEAS-DTC'08) that took place in Edinburgh (Scotland) on July 2008 (Patrón and Lane, 2008).

- The evaluation trials were described in the paper *Fault tolerant adaptive mission planning with semantic knowledge representation for autonomous underwater vehicles* presented at the **IEEE/RSJ International Conference on Intelligent RObots and Systems** (IROS 2008) in Nice (France) on September 2008 (Patrón et al., 2008b).

- Results from this chapter were presented in the paper *Adaptive mission plan diagnosis and repair for fault recovery in autonomous underwater vehicles* during the **International Conference IEEE Oceans** in Quebec (Canada) on September 2008 (Patrón et al., 2008c).

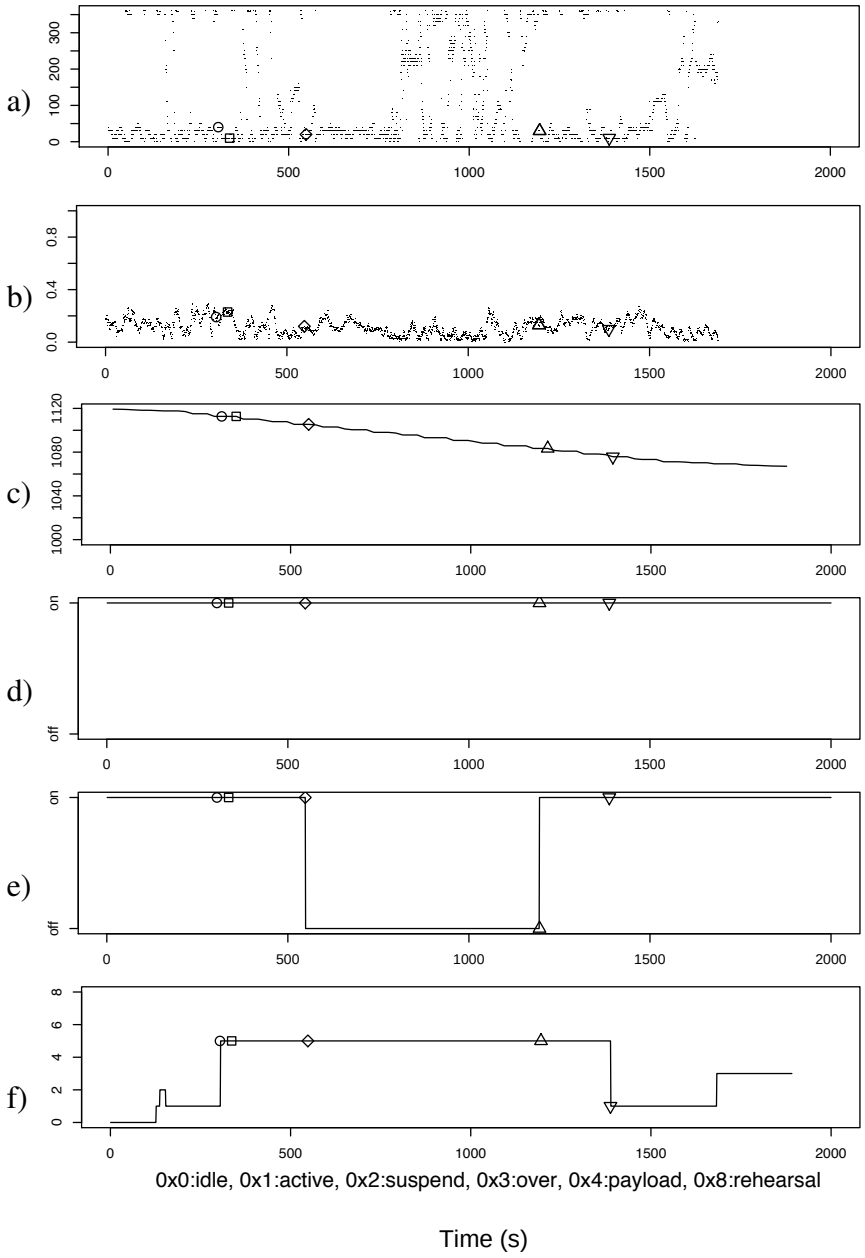


Figure 6.11: Status monitoring (top to bottom): a) direction of water current (degrees), b) speed of water current (m/s), c) battery power (Wh), d) sidescan sensor port and e) starboard transducers availability (on/off) and f) mission status binary flag, all plotted against mission time (s).

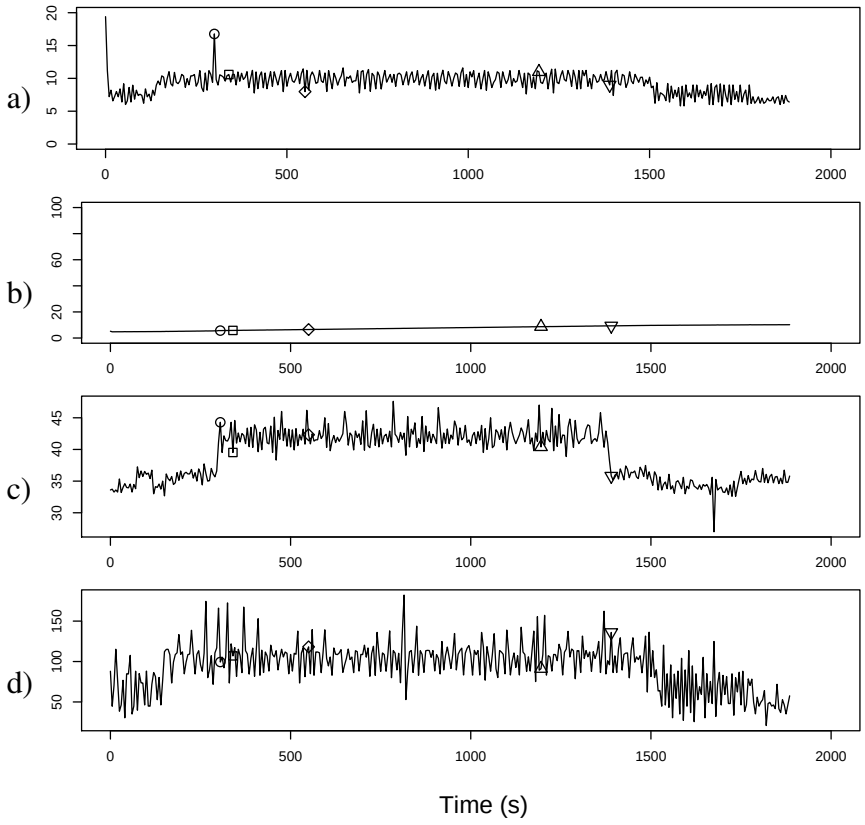


Figure 6.12: System activity (top to bottom): a) % processor usage, b) % memory usage, c) network activity (packets/s) and d) disk usage (I/O sectors/s), all plotted against mission time (s).

Chapter 7

Conclusion and Future Work

Where there is no hope in the future, there is no power in the present.

– John Maxwell

7.1 Introduction

Over the course of this thesis we have introduced several approaches for solving the different requirements for autonomous decision making for UUVs. Each of these approaches targeted one or more of the research objectives introduced at the beginning of this thesis.

In this chapter, we review the contributions and key results achieved by this thesis. Then, we present a set of extensions that show the direction that this research could take in the future. Finally, we highlight a set of projects and programs that are already exploiting the work developed in this research.

7.2 Conclusion

This section presents a review of the contributions, findings, and key results achieved by this research. At the beginning of this thesis, we discussed how marine applications are gaining routine and permanent access to the challenging underwater domain with the use of robotic platforms. We highlighted how these underwater robots currently rely upon waypoint-based scripted missions which are described by the operator a-priori. This leaves vehicles performing data gathering tasks, where sensor data is processed off-line. These two factors make actual systems incapable of reacting to the

unexpected. As a consequence, this severely limits their ability to operate reliably in the very unpredictable domain of underwater. We identified that this challenging environment requires the development and integration of embedded intelligence that can raise autonomy levels whilst maintaining the trust of the operator.

Based on an analysis of several scenarios related to the oceanography, energy and military disciplines, we focused our research on five main objectives:

- The main focus of this research was to investigate mission plan adaptability techniques for autonomous decision making in UUVs. We expected embedded intelligent agent capabilities providing on-board real time autonomous decision making to improve UUVs operability and rates of mission success.
- This adaptability at the mission level required a machine-understandable representation of the environment. This required an analysis of techniques which can bring situation awareness to embedded solutions, via abstracting away from the raw sensor data to conceptual information and linking to the prior knowledge gathered by the operator in previous experience.
- Underwater robots are systems composed of multi-disciplinary payloads and sensors. We needed to investigate which planning techniques were most suitable for autonomously determining the mission plan inside a discoverable environment of published services.
- We required the solution to be robust and portable: robust in balancing time-consuming mission planning, with rapid response reactive actions; scalable and portable to different platforms in different domains.
- Ultimately, in order to quantify the increase in operability, we needed to study methods of measuring improvements in mission adaptability.

In meeting these objectives, this research provided the following novel contributions:

1. After analysing current approaches for autonomous decision making, we identify two problems affecting the effectiveness of the decision loop. Firstly, orientation and observation should be linked together because it is desirable to place the new observations in context. Secondly, decision and action should be iterating continuously. We justify how these elements are required to perform together in order to create an autonomous decision making framework for UUVs.

Under this framework of observation, orientation, decision and action stages we implemented three different approaches to goal-based mission planning in order to investigate which mission planing approach was most appropriate under different circumstances. We found goal-based declarative mission planning to be an attractive solution to autonomous adaptability, because it combines autonomous decision making with higher levels of human interaction. The first approach, continuous mission planning, focusses on long-term deployment using a Markov decision process framework that accommodates fixed-length searches over the state space. The second, service-oriented mission planning, looks at using the dynamic discovery of platform capabilities to guide the selection of states during a backward-chain state-space search. The last approach, mission plan repair, handles small mission environment changes under limited computational resources by balancing adaptability between the planning and the execution levels.

2. These goal-based planning approaches require the use of high-level knowledge representation for user interaction and situation awareness. Situation awareness is critical to link prior knowledge provided by the operator and previous experience with the information coming from the processed sensor data. After analysing relevant approaches, we identified that this was unavailable in current autonomous platforms.

   In order to achieve this, we proposed a semantic-based knowledge representation framework. The framework uses a hierarchy of ontologies to integrate the initial expert orientation and the observations acquired during mission. This provides semantic knowledge interoperability among all information sources involved in a mission. Thus, it improves local (service level) and global (system level) unmanned situation awareness: it allows different embedded agents to communicate while at the same time remaining independent of each other, responsible only for the services that they are providing. These services can also contribute to the world model, enriching the knowledge available to all the agents. It also includes a reasoning interface for inferring new knowledge from the observed data and knowledge stability by checking for inconsistencies. This allows agents to abstract away from the raw real-world data and the host platform in a step by step fashion, leveraging the power of semantic technologies.

   We demonstrated the advantages of the framework by describing the interac-

tion between the status monitor and the planner agents performing a real world scenario. This scenario showed how the framework can act as an enabler for autonomy and on-board decision making. Also, our approach liberates the operator from the low-level understanding of the platform functionalities and makes it easier for the operator to interact with the platform to describe and monitor the mission. As a consequence, it ultimately increases trustworthiness.

Currently we have only integrated the semantic-framework using two agents, but it is extensible to more. It is also a centralised framework on a single platform, but there are no theoretical obstacles to distributing this over multiple platforms. This would allow agents residing on other platforms to contribute to the knowledge enrichment of the mission environment.

3. In order to evaluate our approach, we analysed metrics measuring the adaptation process. After identifying the drawbacks of these metrics, we introduced a novel metric, Plan Proximity, which measures the proximity between plans. We proved that this metric is better informed than previous metrics because it correlates better with the degree of changes that we introduce in a plan. With this metric we show that we can quantify the amount of change introduced by adaptive mission planning.

   There is a clear benefit to having a unique final value for comparison of plans and plan strategies. However, by balancing Plan Difference and State Difference to obtain Plan Proximity, relevant information gets diluted. Under these situations, the selection of the adequate balance factor for a given domain can become critical. As setting this parameter is domain dependant, we prefer to report the individual plan and state differences as well as the Plan Proximity score.

   Plan Proximity does not capture domain-specific differences which depend on rich semantic knowledge. As a consequence, we extended our metric to capture semantic information. By making use of some valid assumptions about the domain model, we were able to maintain the semantic metric independent from any domain-specific measurements. Plan Proximity is slightly more strongly correlated with the amount of change introduced than Semantic Plan Proximity. However, the quantity of change is not as meaningful as the types of changes made: Semantic Plan Proximity is more informed than Plan Proximity. Ideally we would want human judgements to conclusively evaluate the advantage of Semantic Plan Proximity.

4. After analysing current approaches to adaptive mission planning, we proposed a novel approach specifically suited to UUVs operating in dynamic and uncertain discoverable mission environment. We have assumed that the information provided by the knowledge base is fully observable to the planner, i.e. the uncertainty arising from sensor limitations is handled by the agents processing lower-level data. We have also assumed that the mission environment is dynamic and uncertain, i.e. events occur and actions do not always perform as expected.

   Under these assumptions, our approach implements a Bayes paradigm for prediction, measurement, and correction inside a Markov decision process mathematical framework. Based on a continuous re-assessment of the status of the mission environment, our approach provides a decision making loop capable of adapting mission plans. Instead of solving a plan from initial state to goals like in classical AI planning, we maintain a finite size plan that it is believed can be performed from the current state in order to improve a given utility function. Markov decision processes are normally implemented in robotics for low-level control or trajectory planning. In our case, we implemented an approach for mission planning. The goal of planning under this framework is to identify the policy that maximises the expected cumulative payoff. These policies are generally mappings from state to control actions. In our approach, we map from state to plan candidates. This means that we implement a policy that balances the selection of plans by using their estimated cost of execution and the reward obtained by reaching the new configuration of the mission environment.

   We looked at the cumulative payoff over a fixed window length. We adopted the value iteration model to implement the search policy, finding the best policy using classical state-space search. This combination of a finite approach with classical search allowed computational efficiency: by keeping track of the planned actions, we are able to forecast the impact of sensed events in the precomputed plan and, if necessary, react in advance. Given a planning horizon (window size), the solution provided by the approach is optimal for a greedy behaviour and pseudo-optimal for a lazy behaviour. In summary, our implementation is able to handle temporal planning with durative actions, metric planning, opportunistic planning and dynamic planning.

   Using the Plan Proximity metric, the approach was evaluated under a static scenario and a partially known dynamic scenario. The comparison results showed a

high degree of similarity between our approach and the humanly driven adaptation. This indicated that the approach can be trusted by an operator as it provided similar outputs. During this exercise we found difficulties in assigning the right rewards and costs for the different elements of the domain. As a consequence, some of the outputs provided mission plans with unexpectedly high rewards under very myopic conditions. Future learning techniques are expected to assist us in extracting these values from the domain expert. Also, other search alternatives to maximise the average payoff instead of total, infinite horizons where the discount factor may have more impact, and stochastic partial-observability are being considered.

5. We analysed mission planning under a service-oriented architecture. We identified that the total set of actions of a system is the union of all the functionalities provided by all the services in the system. On this basis, we further applied the semantic framework to a goal-based approach capable of providing autonomous mission planning for the dynamic discovery of the services published by the different agents running in the system. This goal-based approach is based on backward state-space search from the mission objectives described by the operator to the current state of the mission environment. By distributing the selection of state candidates during the search process, this approach can be applied to mission plan generation in a service discovery framework. This allowed the software to be both platform independent, easing the manual creation of mission plans, and to be robust to dynamic changes in the platform configuration. This permits autonomous adaptive mission planning when the capabilities of the platform are not known a-priori.

   The approach was evaluated under a service discoverable scenario using a series of communication messages to publish and report the capabilities of the available agents. Results showed that this discovery-based implementation finds the same results as the baseline which was explicitly provided with the platform configuration. The results obtained impact on mission flexibility, robustness and autonomy.

   Due to the communication messages transmitted over the network, our implementation took longer than the original baseline to compute the mission plan. However, if a human was required to configure each change it would take a great deal longer. The approach used in the experiments implemented the challeng-

ing situation in which the capabilities need to be published continuously during the planning process. This contemplates the case where the time of planning of a mission plan and the time of executing it are similar and the system components are very unreliable. In a practical scenario, this approach can be simplified by requesting publication of capabilities from the services only at the beginning of each planning process. This would reduce the communication needs while maintaining the capability of dynamic discovery of actions.

6. We identified how platforms have limited computational resources and how generating a mission plan is usually a time consuming task. Regeneration of the mission plan when changes occur is not always possible. In situations with constantly changing conditions, planning from scratch again can be too slow. Also, the previous effort spent planning is wasted.

   We compare replanning, or plan regeneration, and plan repair techniques. Plan repair is capable of modifying already available plans to cope with the new environmental conditions. We proposed a mission plan repair implementation based on a partial mission plan representation using an iteration of unrefinement and refinement techniques. Unrefinement is used to remove the constraints affecting the current plan under the new conditions. Refinement is then used to add the new constraints required to cope with the new conditions. By using the Plan Proximity metric, we showed that repairing a mission plan tends to provide closer solutions to the original mission plan than replanning. We also looked at minimising the plan repair by providing mission repair at the executive level. We achieve this by making use of alternative instantiations of the mission plan actions.

   Plan repair is desirable when the changes are limited in scope. At some point, with a major change in the environment, it is preferable to replan. In previous research on trajectory planning (Patrón et al., 2007b), we analysed the point at which repairing becomes more computationally expensive than replanning. It would be desirable to do this for mission planning as well.

   Finally, we developed and deployed an implementation of a semantic-based autonomous planning system in a real underwater vehicle. This system combines a semantic-based hierarchical representation of knowledge and mission plan adaptation techniques. The advantage of this combination is that it maximises robustness, system performance and response time. By making use of an abstraction

layer interface, the approach is platform independent making it readily applicable to other domains, such as ground or air vehicles.

This is the first time that an approach to goal-based planning is applied to the adaptation of an underwater mission in order to maintain platform's operability. The results showed adaptability to environmental elements, such as water current flows in order to improve mission performance. The approach was also capable of dealing with changes in status of certain components in the platform and was able to react accordingly.

With these results, we claim that all the research objectives have been accomplished. Our hypothesis stated that the ability to autonomously adapt the decision making process is the key to facilitating the change over from human intervention to intelligent autonomy. We have proved our hypothesis by demonstrating two things. Firstly, with adaptive mission planning we have shown that there is no need to keep the human in the decision making loop. Secondly, by providing higher-levels of interaction with the operator, and by generating plans which are similar to human generated plans, we have made the platforms more accessible and trustworthy for human operators.

Considering the experience gained during this research, the difficulties evaluating our algorithms on real platforms in the underwater domain were striking. Underwater platforms are scarce and trials are expensive. Our efforts to demonstrate functionality at the highest level of the autonomy pyramid have required a monumental effort of engineering, as we depended on all the simpler elements to be running in a reliable manner.

As a consequence, we were unable to provide a thorough comparison between the different approaches. However, based on the potential that they show, we favour the continuous mission planning approach. This approach is more flexible and extensible than the others. Its implementation has shown the potential to handle different types of planning, including opportunistic planning for exploration, a very desirable property for long-term deployments of UUVs. Under the semantic-based framework, we think that the action selection of this approach can be extended to deal with the discovery of service-oriented capabilities handled by the second approach. Mission plan repair requires an initial mission plan. This is not always possible when the environment is largely unknown. On the other hand, plan repair is more predictable than continuous planning as it provides a more limited adaptation, and is therefore more easily trusted by the operator. Continuous mission planning is also myopic to the planning hori-

zon. This limits the execution of tasks requiring a longer sequence of actions than the planning horizon. Another drawback is that the continuous mission planning search for optimal policies is polynomial in the size of the state space. However, this space grows exponentially with the discovery of new knowledge. Several techniques are being considered to try to reduce the part of the state space that is being considered (Dean et al., 1993; Boutilier and Dearden, 1994). In the future, we are planning to work towards incorporating the benefits of the service-oriented architecture and the two levels of mission plan repair into the continuous mission planning approach to make it more efficient and more generic.

On this journey we have encountered two very separate communities: the planning and the ocean engineering community. We think that the planning community should focus more on the practical application of their work and not be so exclusively focussed on theoretical, and often impractical, improvements in optimality. Also, the ocean engineering community should look carefully at the thoughtful science coming from the AI community. Some of the work there is ready to be incorporated into their systems and to be trusted by the operators. We straddled both fields and this was very productive.

We found openness in the architecture of the platform to be critical for practical implementations. We are pleased to find that the open architecture model is now widely accepted and is slowly becoming available. Equally, we are proud to be part of the effort behind this movement.

We showed how all other underwater challenges for robotics are now being mitigated by a level of maturity achieved by their correspondent technologies. We think that the moment in time has come to delegate tedious underwater tasks to these robotic platforms. This will ultimate make platforms more reliable and more accessible to a wider community. However, in this process, we predict that essential metrics to validate the systems and their autonomy will become a priority for the ultimate acceptance by the final customer.

The work presented here is just a small contribution to the field. However, these initial findings are now ready to be taken further, to a series of future extensions. The future work is presented in Section 7.3. There is huge demand for this technology to mature in order to fulfil the needs of the user. In Section 7.4 we describe the planned exploitations of this research.

7.3 Future Work

This section describes possible extensions to the research presented in this thesis.

7.3.1 Shared Situation Awareness

In Section 2.3.2 of Chapter 2, we introduced a semantic-based framework for a system with a single platform. The framework integrates orientation and observation for providing unmanned situation awareness. This framework allows knowledge distribution between embedded agents at all levels of representation. The framework also enables autonomous discovery of system capabilities as part of the service-oriented mission planning approach proposed in Chapter 5.

For the next stage, we will be looking at extensions to the framework that will allow the distribution of knowledge between a group or team of vehicles. We are currently working towards providing situation awareness for a team of vehicles in which every team member possess the required awareness for its responsibilities (Cartwright et al., 2007). The main challenge here is to deal with the acoustic communication limitations associated with the underwater environment (see Section 1.2.3.2 for a detailed description of underwater communication limitations).

7.3.2 Human-based Evaluation of Semantic Plan Proximity

In Chapter 3, we introduced a metric formulation for measuring the performance of planning strategies. The extension of the metric presented in Section 3.5.2.8 incorporated semantic knowledge about the domain. By making use of two basic assumptions, hierarchy of the classes in the domain and semantic exclusivity of the propositions, the metric remained independent from any domain-specific measurements.

Semantic Plan Proximity has been shown to be more informed than Plan Proximity as it considered the type of change as well as the quality of change. However, Plan Proximity was slightly more strongly correlated to the amount of change. It will be necessary to undertake a human evaluation experiment in order to validate this metric with the views of a human expert. The use of crowdsourcing techniques is being explored (Crowdsourcing, 2010; Mechanical Turk, 2010).

7.3.3 Machine Learning of Costs and Rewards

In Section 4.4 of Chapter 4 we introduced an approach to continuous mission planning that handles temporal planning with durative actions, metric planning, opportunistic planning and dynamic planning. During the decision making process, the plan selection was balanced between its estimated cost of execution and the reward obtained by reaching the new configuration of the environment.

In order to obtain mission plans that perform as desired by the operator, the estimation of costs and rewards for a new mission environment requires a training phase capable of finding the adequate weights and parameters for the algorithms. The human evaluation of the semantic-based metric proposed in Section 7.3.2 could create a corpora that sets the basis from which these parameters can be automatically learned.

7.3.4 Hierarchical Timelines

In Section 5.3.2 of Chapter 5 we demonstrated how independent service-oriented agents can coordinate themselves with a goal-based planner to find a mission plan that covers the mission requirements and that agrees with the available capabilities of the platform. The approach proposed in Section 4.4 of Chapter 4 for adaptive continuous mission planning maintains a single timeline of execution and it is limited by the maximum planning horizon that can be achieved for the given computational resources.

A natural extension will be to create a hierarchy of timelines from where it will be possible to plan at different levels of granularity for the different service-oriented agents. A similar approach has been implemented in T-REX architecture using reactive deliberators (McGann et al., 2009).

7.3.5 Distributed Mission Planning

A shared situation awareness framework, like the one proposed in Section 7.3.1, enables distributed mission planning for multi-platform systems.

Most of previous research work on cooperative underwater vehicles has been focused on behavioural approaches that do not explicitly reason about assigning goals and planning courses of action. Others are just reactive planning techniques incapable of mission planning with high-level goals in an efficient manner.

Due to the communication limitations of the domain, we think that cooperation for multi-platform underwater systems should be addressed by an approach that im-

plements self-interested, benevolent mission planning agents for each platform. Under such approach, vehicles would have their own private goals but they will be willing to cooperate to help other vehicles in their tasks as long as their own private goals are not affected. Under these assumptions, we believe that the approach proposed in Chapter 4 for continuous mission planning can be extrapolated to cooperative platforms using a timeline for each of the platforms and some of the principles for cooperative mission execution that we shown in Evans et al. (2006).

7.4 Exploitation

Four main opportunities for exploiting this research are being undertaken over the next few years. These are:

- The MoD's DTIC Competition of Ideas technology contract is currently sponsoring the project 'Semantic World Modelling for Distributed Knowledge Representation in Co-operative (Autonomous) Platforms' (RT/COM/5/059). This project is the result of the findings on semantic-based knowledge representation for unmanned vehicles presented in Chapter 2.

- The MoD's SEAS-DTC Integrated Demonstration Group lead by BAE Systems is looking at exploitation of technologies capable of providing an integrated solution for mine countermeasures using underwater platforms, and cross domain operations using a heterogeneous team of ground and air vehicles. The work developed under this research has been selected as one of these technologies. The outcomes from Chapter 4 are part of the solution proposed for the Exemplar I (underwater domain) demonstration.

- The MoD's DSTL Osprey consortium, involving BAE Systems, SEA Ltd. and Heriot-Watt University, will be looking at autonomy for underwater mine countermeasure operations using unmanned platforms. The outcomes of this research in performance metrics, awareness and autonomy are being considered to form part of the solution for intelligent search and collaborative platforms.

- The Student Autonomous Underwater Competition - Europe (SAUC-E) will take place at the NATO Underwater Research Centre in La Spezia, Italy. The challenge proposed to the teams requires high levels of mission adaptability and capability to react to environmental events. The outcomes of this research are

being integrated to provide the high level control to the vehicle of the team from Heriot-Watt University.

Additionally, Heriot-Watt University is planning to exploit the outcomes of this project through its spin out company SeeByte Ltd. which has market access and credibility in the field having deployed 100+ situation awareness tools for the US and other Navies in the subsea Mine and Counter Measure Arena. SeeByte Ltd. is a growing SME which has the capability to move this research to the market.

Appendix A

Proof of Distance of Plan Proximity

A.1 Metrics

Definition A.1.1 *A metric on a set $\mathcal{X}$ is a function (called the distance function or simply distance)*

$$d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \tag{A.1}$$

For all $x, y, z \in X$, this function is required to satisfy the following conditions:

1. *Non-negativity, i.e. $d(x,y) \geq 0$*
2. *Identity of indiscernibles*[1]*, i.e. $d(x,y) = 0$ if and only if $x = y$*
3. *Symmetry, i.e. $d(x,y) = d(y,x)$*
4. *Subadditivity (a.k.a. triangle inequality), i.e. $d(x,z) \leq d(x,y) + d(y,z)$*

The edit distance between two strings of characters is the number of operations required to transform one of them into the other.

The 'diff' algorithm defines an edit distance by solving the longest common subsequence (LCS) problem and taking the ordering of the elements into account. The 'diff' algorithm is a distance metric (Hunt and McIlroy, 1976).

The Plan Difference $\hat{D}_p$ was defined in Section 3.3.2.1. It is applied to the set of all possible mission plans solving any mission environment from a given domain space Θ_π. It is based on the 'diff' algorithm. Thus, it satisfies the aforementioned conditions, $\forall \pi_a, \pi_b, \pi_c \in \Theta_\pi$:

1. $\hat{D}_p(\pi_a, \pi_b) \geq 0$

[1] Note that condition 1 and 2 together produce positive definiteness, i.e. $f(0) = 0 \wedge f(x) > 0,\ \forall x \neq 0$.

2. $\hat{D}_p(\pi_a, \pi_b) = 0 \iff \pi_a = \pi_b$
3. $\hat{D}_p(\pi_a, \pi_b) = D_p(\pi_b, \pi_a)$
4. $\hat{D}_p(\pi_a, \pi_b) \leq D_p(\pi_a, \pi_c) + D_p(\pi_c, \pi_b)$

The Hamming distance is another instantiation of an edit distance that does not contemplate the ordering of the elements (Hamming, 1950).

The State Difference $\hat{D}_s$ was defined in Section 3.3.2.2. It is applied to the set of all possible states of a given domain space Θ_s. It is based on the Hamming distance. Thus, it also satisfies the distance properties, $\forall s_a, s_b, s_c \in \Theta_s$:

1. $\hat{D}_s(s_a, s_b) \geq 0$
2. $\hat{D}_s(s_a, s_b) = 0 \iff s_a = s_b$
3. $\hat{D}_s(s_a, s_b) = \hat{D}_s(s_b, s_a)$
4. $\hat{D}_s(s_a, s_b) \leq \hat{D}_s(s_a, s_c) + \hat{D}_s(s_c, s_b)$

A.2 Demonstration

Plan Proximity has been defined in Section 3.3.2.3 as:

$$\begin{aligned} PP_\alpha(\pi_1, \pi_2) &: (\Theta_\pi, \Theta_s) \times (\Theta_\pi, \Theta_s) \rightarrow [1; 0] \\ &= 1 - \alpha \cdot \hat{D}_p(\pi_1, \pi_2) - (1 - \alpha) \cdot \hat{D}_s(s_1, s_2) \end{aligned}$$

where s_1 and s_2 are the final states obtained after the execution of π_1 and π_2 respectively.

The complement of Plan Proximity is defined as:

$$\begin{aligned} \bar{PP}_\alpha(\pi_1, \pi_2) &: (\Theta_\pi, \Theta_s) \times (\Theta_\pi, \Theta_s) \rightarrow [0; 1] \\ &= 1 - PP_\alpha(\pi_1, \pi_2) \end{aligned}$$

Based on the previous assumptions, it can be demonstrated that the complement of Plan Proximity satisfies the distance properties. For any given balance factor α, triple of mission plans $\forall \pi_1, \pi_2, \pi_3 \in \Theta_\pi$ and their correspondent final states $\forall s_1, s_2, s_3 \in \Theta_s$ the following metric distance properties are satisfied:

1. Non-negativity

$$\begin{aligned} \bar{PP}_\alpha(\pi_1, \pi_2) &= \alpha \cdot \hat{D}_p(\pi_1, \pi_2) + (1 - \alpha) \cdot \hat{D}_s(s_1, s_2) \\ &\geq \alpha \cdot 0 + (1 - \alpha) \cdot 0 \\ &\geq 0 \end{aligned}$$

2. Identity of indiscernibles:

$$\begin{aligned} \bar{PP}_\alpha(\pi_1,\pi_1) &= \alpha \cdot \hat{D}_p(\pi_1,\pi_1) + (1-\alpha) \cdot \hat{D}_s(s_1,s_1) \\ &= \alpha \cdot 0 + (1-\alpha) \cdot 0 \\ &= 0 \end{aligned}$$

3. Symmetry:

$$\begin{aligned} \bar{PP}_\alpha(\pi_1,\pi_2) &= \alpha \cdot \hat{D}_p(\pi_1,\pi_2) + (1-\alpha) \cdot \hat{D}_s(s_1,s_2) \\ &= \alpha \cdot \hat{D}_p(\pi_2,\pi_1) + (1-\alpha) \cdot \hat{D}_s(s_2,s_1) \\ &= \bar{PP}_\alpha(\pi_2,\pi_1) \end{aligned}$$

4. Triangle Inequality:

$$\begin{aligned} \bar{PP}_\alpha(\pi_1,\pi_2) &= \alpha \cdot \hat{D}_p(\pi_1,\pi_2) + (1-\alpha) \cdot \hat{D}_s(s_1,s_2) \\ &\leq \alpha \cdot (\hat{D}_p(\pi_1,\pi_3) + \hat{D}_p(\pi_3,\pi_2)) + (1-\alpha) \cdot (\hat{D}_s(s_1,s_3) + \hat{D}_s(s_3,s_2)) \\ &\leq \alpha \cdot \hat{D}_p(\pi_1,\pi_3) + (1-\alpha) \cdot \hat{D}_s(s_1,s_3) + \alpha \cdot \hat{D}_p(\pi_3,\pi_2) + (1-\alpha) \cdot \hat{D}_s(s_3,s_2) \\ &\leq \bar{PP}_\alpha(\pi_1,\pi_3) + \bar{PP}_\alpha(\pi_3,\pi_2) \end{aligned}$$

Therefore $\bar{PP}_\alpha$ is a distance metric. Thus, PP_α is a distance-based metric, as is the complement of a distance metric. It looks at the proximity between elements rather than at the distance between them.

Appendix B

Example of a mission environment scenario

B.1 Domain model

```
1  ; Domain : sauce_003
2  ; Author : $Author: patron $
3  ; Date   : $Date: 2010/01/20 16:13:22 $
4  ; Group  : Example for continuous planning
5  ;
6  (define (domain sauce_003)
7  ;-----------------------
8  ; REQUIREMENTS
9  ;-----------------------
10   (:requirements
11     :typing
12     :fluents
13     :durative-actions
14     :world-modeling
15     :continuous
16   )
17 ;-----------------------
18 ; TYPES
19 ;-----------------------
20   (:types
21     location
22     area - location
23     gate - location
24     wall - location
```

```
25      dock - location
26      vehicle - object
27      target - physob
28      sensor - object
29      light - object
30      position - object
31      direction - object
32      situation - object
33      camera - sensor
34      status - object
35    )
36 ;-----------------------
37  ; CONSTANTS
38 ;-----------------------
39    (:constants
40      red green off - light
41      up down - direction
42      left right centre - position
43      engaged detached - status
44    )
45 ;-----------------------
46  ; PREDICATES
47 ;-----------------------
48    (:predicates
49      (at ?l - location)
50      (colored ?g - gate ?l - light)
51      (colreg ?l - light ?p - position)
52      (moving ?t - target)
53      (still ?t - target)
54      (docked ?s - status)
55      (facing ?d - direction)
56      (passed-up  ?g - gate ?p - position)
57      (passed-down ?g - gate)
58      (visited ?l - location)
59      (inspected ?s - sensor ?t - target)
60      (tracked ?s - sensor ?t - target)
61      (surveyed ?s - sensor ?a - wall)
62    )
63 ;-----------------------
64  ; FUNCTIONS
65 ;-----------------------
66    (:functions
```

```
67      (distance ?from ?to - location)
68      (consumption ?s - sensor)
69    )
70 ;-----------------------
71 ; ACTIONS
72 ;-----------------------
73    (:passive-action toWait
74      :duration    (= ?duration 1)
75    )
76    (:durative-action toMove
77      :parameters    (?from ?to - location)
78      :duration    (= ?duration (distance ?from ?to))
79      :condition     (and
80                (at start (at ?from))
81                (at start (docked detached))
82              )
83      :effect      (and
84                (at end (not (at ?from)))
85                (at end (at ?to))
86                (at end (visited ?to))
87              )
88    )
89    (:durative-action toTraverse-in
90      :parameters    (?g - gate ?l - light ?p - position)
91      :duration    (= ?duration 2)
92      :condition     (and
93                (at start (at ?g))
94                (at start (facing up))
95                (at start (colored ?g ?l))
96                (at start (colreg ?l ?p))
97              )
98      :effect      (and
99                (at end (passed-up ?g ?p))
100             )
101   )
102   (:durative-action toTraverse-out
103     :parameters    (?g - gate)
104     :duration    (= ?duration 2)
105     :condition     (and
106               (at start (at ?g))
107               (at start (facing down))
108             )
```

```
109     :effect      (and
110               (at end (passed-down ?g))
111             )
112   )
113   (:durative-action toTurn
114     :parameters   (?d1 - direction ?d2 - direction)
115     :duration    (= ?duration 2)
116     :condition     (and
117               (at start (facing ?d1))
118             )
119     :effect      (and
120               (at end (not (facing ?d1)))
121               (at end (facing ?d2))
122             )
123   )
124   (:durative-action toSurvey
125     :parameters   (?s - sensor ?w - wall)
126     :duration    (= ?duration 30)
127     :condition     (at start (at ?w))
128     :effect      (and
129               (at end (surveyed ?s ?w))
130             )
131   )
132   (:durative-action toInspect
133     :parameters   (?s - sensor ?t - target)
134     :duration    (= ?duration 10)
135     :condition     (and
136               (at start (at ?t))
137               (at start (still ?t))
138             )
139     :effect      (and
140               (at end (inspected ?s ?t))
141             )
142   )
143   (:durative-action toFollow
144     :parameters   (?s - sensor ?t - target)
145     :duration    (= ?duration 10)
146     :condition     (and
147               (at start (at ?t))
148               (at start (moving ?t))
149             )
150     :effect      (and
```

```
151                (at end (tracked ?s ?t))
152              )
153    )
154    (:durative-action toDock
155      :parameters    (?d - dock)
156      :duration    (= ?duration 1)
157      :condition     (and
158                (at start (at ?d))
159                (at start (docked detached))
160              )
161      :effect      (and
162                (at end (docked engaged))
163              )
164    )
165 )
```

B.2 Problem model

```
1  ; Problem      : sauce_003
2  ; Domain       : sauce_003
3  ; Author       : $Author: patron $
4  ; Date         : $Date: 2010/01/20 16:13:22 $
5  ; Description : Example for continuous planning
6  ;
7  (define (problem sauce_003)
8  ;-----------------------
9  ; DOMAIN
10 ;-----------------------
11   (:domain sauce_003)
12 ;-----------------------
13 ; OBJECTS
14 ;-----------------------
15   (:objects
16   )
17 ;-----------------------
18 ; INIT STATE
19 ;-----------------------
20   (:init
21     (docked detached)
22     (at lstart)
```

```
23      (facing up)
24      (colored gate1 off)
25      (colreg red right)
26      (colreg green left)
27      (colreg off centre)
28    )
29 ;-----------------------
30 ; REWARDS
31 ;-----------------------
32    (:reward
33      (= lstart 0)
34      (= gate1 3)
35    )
36 ;-----------------------
37 ; GOALS
38 ;-----------------------
39    (:goal
40 ;    (= (visited ?l) 10)
41      (= (passed-up ?g ?p) 140)
42 ;    (= (passed-up ?g centre ?d) 100)
43      (= (passed-down ?g) 70)
44      (= (inspected downward ?t) 50)
45      (= (tracked forward ?t) 50)
46      (= (surveyed forward ?w) 50)
47      (= (docked engaged) 20)
48    )
49 ;-----------------------
50 ; METRIC
51 ;-----------------------
52    (:metric minimize (total-duration))
53    )
54 )
```

B.3 World model

```
1 ; World Model : $RCSfile: sauce_003.wm,v $
2 ; Problem     : sauce_003
3 ; Domain      : sauce_003
4 ; Author      : $Author: patron $
5 ; Date        : $Date: 2010/01/20 16:13:22 $
```

```
6  ; Description : Example for continuous planning
7  ;
8  (define (world sauce_003)
9  ;-------------------------
10 ; DOMAIN
11 ;-------------------------
12 ; (:domain sauce_004)
13 ;-------------------------
14 ; PROBLEM
15 ;-------------------------
16 ; (:problem sauce_004)
17 ;-------------------------
18 ; NUMBER
19 ; Name, Value
20 ;-------------------------
21   (:number distance-traveled     0.0)
22 ;-------------------------
23 ; VEHICLE
24 ; Name, ID, Distance-Traveled, Battery, Speed
25 ;-------------------------
26 ; (:vehicle remus337    337    0.0    90.0   1.5)
27 ;-------------------------
28 ; LOCATION
29 ; Name, Lat, Lon, Depth
30 ;-------------------------
31   (:location lstart    56.38400   -4.27010   4)
32 ;-------------------------
33 ; SENSOR
34 ; Name, Vehicle, Powered, Consumption
35 ;-------------------------
36   (:sensor video         337 on   8)
37   (:sensor sidescan        337 on   12)
38 ;-------------------------
39 ; CAMERA
40 ; Name, Vehicle, Powered, Consumption, FocalLength, Aperture
41 ;-------------------------
42   (:camera forward         337 on 3 35 1.8)
43   (:camera downward        337 on 3 35 1.8)
44 ;-------------------------
45 ; GATE
46 ; Name, Lat, Lon, Depth, Diameter
47 ;-------------------------
```

```
48    (:gate gate1     56.38410   -4.27010   4 20)
49 ;-------------------------
50 ; DOCK
51 ; Name, Lat, Lon, Depth, Width, Height
52 ;-------------------------
53    (:dock recovery 56.38440   -4.27000   4 5 5)
54 )
```

B.4 Dynamic world model

```
1  ; World Model : $RCSfile: sauce_003.dwm,v $
2  ; Problem     : sauce_003
3  ; Domain      : sauce_003
4  ; Author      : $Author: patron $
5  ; Date        : $Date: 2010/01/20 16:13:22 $
6  ; Description : Example for continuous planning
7  ;
8  (define (dynamic-world sauce_003)
9     (:domain sauce_003)
10    (:add (:events  (at gate1))
11        (:objects (:gate gate2     56.38420   -4.27010   4 20))
12        (:effect  (colored gate2 green))
13    )
14    (:add (:events  (visited gate1))
15        (:objects )
16        (:effect  (not (colored gate2 green))
17              (colored gate2 red))
18    )
19    (:add (:events  (visited gate2))
20        (:objects (:gate gate3     56.38430   -4.27010   4 20))
21        (:effect  (colored gate3 off))
22    )
23    (:add (:slot     10)
24        (:objects (:target bottom    56.38405   -4.270020 4))
25        (:effect  (still bottom))
26    )
27    (:del (:slot     12)
28        (:objects (:camera forward))
29        (:action  )
30    )
```

```
31    (:add (:slot     14)
32        (:objects (:target middle    56.38435   -4.270020 4))
33        (:effect  (moving middle))
34    )
35    (:rst (:slot     28)
36        (:objects (:camera forward))
37        (:action  )
38    )
39    (:add (:slot     30)
40        (:objects (:wall wall1   56.38415   -4.270000 4 50   5)
41              (:wall wall2   56.38405   -4.270000 4 50   5)
42              )
43        (:effect  )
44    )
45    (:del (:events  (visited gate2))
46        (:objects )
47        (:action  (toMove))
48    )
49    (:rst (:slot     22)
50        (:objects )
51        (:action  (toMove))
52    )
53 )
```

Bibliography

Ackley, S., Collins, K., Dowdeswell, J., Griffiths, G., and Heywood, K. (2008). AUTOSUB under ice: Combined science use and technical development in a polar programme. In *Scientific Committee on Antarctic Research (SCAR) International Arctic Science Committee (IASC) Polar Research Arctic and Antarctic perspectives in the International Polar Year SCAR/IASC IPY Open Science Conference*, St. Petersburg, Russia.

Adams, J. A. (2007). Unmanned vehicle situation awareness: A path forward. In *Proceedings of the 2007 Human Systems Integration Symposium*, Annapolis, Maryland, USA.

Arkin, R. C. (1998). *Behavior-based Robotics*. Massachusetts Institute of Technology Press, ISBN: 978-0262011655.

Arredondo, M., Reed, S., and Petillot, Y. R. (2006). Battlespace access for unmanned underwater vehicles - dynamic multi-dimensional world modelling - final report. Technical report, SeeByte Ltd. - Ministry of Defence.

Bell, R., Studinger, M., Shuman, C., Fahnestock, M., and Joughin, I. (2007). Large subglacial lakes in east antarctica at the onset of fast-flowing ice streams. *Nature*, 445:904–907.

Bellettini, A. and Pinto, M. A. (2002). Theoretical accuracy of synthetic aperture sonar micro-navigation using a displaced phase-center antenna. *IEEE Journal of Oceanic Engineering*, 27(4):780–789.

Bellingham, J., Consi, T., Beaton, R., and Hall, W. (1990). Keeping layered control simple. In *Proceedings of the (1990) Symposium on Autonomous Underwater Vehicle Technology (AUV'90)*, pages 3–8.

Bellingham, J., Kirkwood, B., and Rajan, K. (2006). Tutorial on issues in underwater robotic applications. In *16th International Conference on Automated Planning and Scheduling (ICAPS'06)*.

Benjamin, M., Curcio, J., Leonard, J., and Newman, P. (2006). Navigation of unmanned marine vehicles in accordance with the rules of the road. *International Conference on Robotics and Automation (ICRA'06)*.

Benjamin, M. R., Leonard, J. J., Schmidt, H., and Newman, P. M. (2009). An overview of MOOS-IvP and a brief users guide to the IvP Helm autonomy software. Technical Report MIT-CSAIL-TR-2009-028, Computer Science and Artificial Intelligence Lab, MIT.

Bennet, A. A. and Leonard, J. J. (2000). A behavior-based approach to adaptive feature detection and following with autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering*, 25(2):213–226.

Bergmann, R. and Wilke, W. (1995). Building and refining abstract planning cases by change of representation language. *J. Artif. Int. Res.*, 3(1):53–118.

Billingham, L. (2004). The role of inspection and intervention AUVs in support of the oil fields of the future. In *Proceedings of the 6th Unmanned Underwater Vehicle Showcase (UUVS'04)*, Southampton, UK.

Blomqvist, E. and Sandkuhl, K. (2005). Patterns in ontology engineering classification of ontology patterns. In *7th International Conference on Enterprise Information Systems*, Miami, USA.

Blum, A. L. and Furst, M. L. (1997). Fast planning through planning graph analysis. In *Artificial Intelligence*, volume 90, pages 281–300.

Boutilier, C. and Dearden, R. (1994). Using abstractions for decision-theoretic planning with constraints. *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 2:1016–1022.

Boyd, J. (1992). A discourse on winning and losing - organic design for command and control. Technical report, http://www.d-n-i.net/dni/john-r-boyd/.

Brooks, R. (1986). A robust layered control system for a mobile robot. *Journal of Robotics and Automation*, RA-2(1):14–23.

Brown, N. (2003). What lies beneath. *Jane's Navy International (ISSN: 13583719)*, 108(5):14–21.

Caccia, M. and Veruggio, G. (2000). Guidance and control of a reconfigurable unmanned underwater vehicle. *Control Engineering Practice*, 8(1):21 – 37.

Caffaz, A., Caiti, A., Casalino, G., Gualdesi, L., and Turetta, A. (2009). Folaga: a low cost AUV/glider for coastal environmental sampling. *Underwater Technology: The International Journal of the Society for Underwater Technology*, 28(4):151–157.

Carbonell, J. G. (1984). *Learning by Analogy: Formulating and Generalizing Plans from Past Experience*. Springer, Berlin, Heidelberg.

Carbonell, J. G. (1993). *Readings in knowledge acquisition and learning: automating the construction and improvement of expert systems*, chapter Derivational analogy: a theory of reconstructive problem solving and expertise acquisition, pages 727–738. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Carreras, M., Palomeras, N., Ridao, P., and Ribas, D. (2007). Design of a mission control system for an AUV. *International Journal of Control*, 80(7):993–1007.

Carreras, M., Ridao, P., Garcia, R., and Batlle, J. (2006). *Behaviour Control of UUVs*, chapter 4, pages 67–86. G. Roberts and R. Sutton, Eds Advances in Unmanned Maritime Vehicles (IEE Control Engineering), ISBN: 978-0863414503.

Cartwright, J., Patrón, P., Evans, J., and Lane, D. M. (2007). Distributed ontological world model for autonomous multi vehicle operations. In *Proceedings of the Conference of Systems Engineering for Autonomous Systems from the Defence Technology Centre (SEAS-DTC'07)*, Edinburgh, UK.

Chang, T. C. and Wysk, R. A. (1984). *An Introduction to Automated Process Planning Systems*. Prentice Hall Professional Technical Reference.

Chardard, Y. and Copros, T. (2002). SWIMMER: final sea demonstration of this innovative hybrid AUV/ROV systems. In *Proceedings of the 2002 International Symposium on Underwater Technology*, pages 17–23.

Chien, S., Knight, R., Stechert, A., Sherwood, R., and Rabideau, G. (1999). Integrated planning and execution for autonomous spacecraft. In *Proceedings of the IEEE Aerospace Conference (IAC'99)*, volume 1, pages 263–271, Aspen, CO, USA.

Chien, S., Knight, R., Stechert, A., Sherwood, R., and Rabideau, G. (2000). Using iterative repair to improve responsiveness of planning and scheduling. pages 300–307.

Chouinard, C., Fisher, F., Gaines, D., Estlin, T., and Schaffer, S. (2003). An approach to autonomous operations for remote mobile robotic exploration. In *Proceedings of the IEEE Aerospace Conference (IAC'03)*, volume 1, pages 1–322, Big Sky, MT, USA.

Collett, T. H., MacDonald, B. A., and Gerkey, B. P. (2005). Player 2.0: Toward a practical robot programming framework. *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005), Sydney, Australia*.

Crowdsourcing (2010). Crowdsourcing. http://en.wikipedia.org/wiki/Crowdsourcing.

Curtin, T. B., Bellingham, J. G., Catipovic, J., and Webb, D. (1993). Autonomous oceanographic sampling networks. *Oceanography*, 6:8694.

Damerau, F. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7:3:171–176.

Davis, B. C., Patrón, P., Arredondo, M., and Lane, D. M. (2007a). Augmented reality and data fusion techniques for improved testing & enhanced situational awareness of the underwater domain. In *Proceedings of the IEEE International Conference Oceans Europe (Oceans Europe'07)*, Aberdeen, UK.

Davis, B. C., Patrón, P., and Lane, D. M. (2007b). An augmented reality architecture for the creation of hardware in the loop and hybrid simulation test scenarios for unmanned underwater vehicles. In *Proceedings of the IEEE International Conference Oceans (Oceans'07)*, Vancouver, Canada.

Dean, T., Kaelbling, L. P., Kirman, J., and Nicholson, A. (1993). Planning under time constraints in stochastic domains. *ARTIFICIAL INTELLIGENCE*, 76:35–74.

Dobeck, G., Hyland, J., and Smedley, L. (1997). Automated detection/classification of sea mines in sonar imagery. *Proceedings of the International Society for Optical Engineering*, 2079:90–110.

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics and Automation Magazine*, 13:99–110.

Endsley, M. R., Bolte, B., and Jones, D. G. (2003). *Designing for Situation Awareness: An Approach to User-centred Design*. Taylor & Francis Group, ISBN: 0-748-40967-1.

Estlin, T., Gaines, D., Chouinard, C., Fisher, F., Castano, R., Judd, M., and Nesnas, I. (2005). Enabling autonomous rover science through dynamic planning and scheduling. In *Proceedings of the IEEE Aerospace Conference (IAC'05)*, volume 1, pages 385–396, Big Sky, MT, USA.

Evans, J., Patrón, P., Privat, B., Johnson, N., and Capus, C. (2009). AUTOTRACKER: Autonomous inspection capabilities and lessons learned in offshore operations. In *Proceedings of the IEEE International Conference Oceans (Oceans'09)*, Biloxi, MS, USA.

Evans, J., Patrón, P., Smith, B., and Lane, D. (2008). Design and evaluation of a reactive and deliberative collision avoidance and escape architecture for autonomous robots. *Journal of Autonomous Robots*, 3:247–266.

Evans, J., Sotzing, C., Patrón, P., and Lane, D. (2006). Cooperative planning architectures for multi-vehicle autonomous operations. In *Proceedings of the Conference of Systems Engineering for Autonomous Systems from the Defence Technology Centre (SEAS-DTC'06)*, Edinburgh, UK.

Farrell, J. A., Pang, S., and Li, W. (2005). Chemical plume tracing via an autonomous underwater vehicle. *IEEE Journal of Oceanic Engineering*, 30, 2:428–442.

Fikes, R. and Nilsson, N. (1971). STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Fisher, F., Knight, R., Engelhardt, B., Chien, S., and Alejandre, N. (2000). A planning approach to monitor and control for deep space communications. In *Proceedings of the IEEE Aerospace Conference (IAC'00)*, volume 2, pages 311–320, Big Sky, MT, USA.

Fitzpatrick, P., Metta, G., and Natale, L. (2007). Towards long-lived robot genes. *Robotics and Autonomous Systems*, 56(1):29–45.

Fossen, T. I. (1994). *Guidance and Control of Ocean Vehicles*. John Wiley and Sons, ISBN: 0-471-94113-1.

Fox, M., Gerevini, A., Long, D., and Serina, I. (2006a). Plan stability: Replanning versus plan repair. In *16th International Conference on Automated Planning and Scheduling (ICAPS'06)*.

Fox, M., Long, D., Baldwin, L., Wilson, G., Wood, M., Jameux, D., and Aylett, R. (2006b). On-board timeline validation and repair: a feasibility study. In *Proceedings of 5th International Workshop on Planning and Scheduling in Space*, pages 22–25, Baltimore, USA.

Fox, M., Long, D., Py, F., Rajan, K., and Ryan, J. (2007). In situ analysis for intelligent control. In *Proceedings of the IEEE International Conference Oceans (Oceans'07)*, Vancouver, Canada.

Frank, J. and Jónsson, A. (2003). Constraint-based attribute and interval planning. volume 8, pages 339–364, Hingham, MA, USA. Kluwer Academic Publishers.

Freitag, L., Grund, M., Singh, S., Partan, J., Koski, P., and Ball, K. (2005). The WHOI micro-modem: An acoustic communcations and navigation system for multiple platforms. In *Proceedings of the IEEE International Conference Oceans (Oceans'05)*, Washington, DC, USA.

Gat, E. (1998). *Artificial Intelligence and Mobile Robots*, chapter On Three-Layer Architectures. D. Kortenkamp, R. Bonnaso, and R. Murphy, Boston, MA, USA.

Gerevini, A. and Serina, I. (2000a). Fast plan adaptation through planning graphs: Local and systematic search techniques. In *Proceedings of the 5th International Conference on AI Planning Systems (AIPS-00)*, pages 112–121, Menlo Park, CA, USA. AAAI Press.

Gerevini, A. and Serina, I. (2000b). Fast plan adaptation through planning graphs: Local and systematic search techniques. In *AAAI Conf. on AI Planning Systems*, pages 112–121.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL: The planning domain definition language. Technical report, Yale Center for Computational Vision and Control.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory and Practice*. Morgan Kaufmann, ISBN: 1-55860-856-7.

Griffiths, G., editor (2003). *Technology and Applications of Autonomous Underwater Vehicles*. Taylor & Francis Group, ISBN 0-415-30154-8.

Griffiths, G. (2005a). AUTOSUB under ice. *Ingenia, ISSN: 1472-9768*, 22:30–32.

Griffiths, G. (2005b). Cost vs. performance for fuel cells and batteries within AUVs. In *Proceedings of the 7th Unmanned Underwater Vehicle Showcase (UUVS'05)*, Southampton, UK.

Gruber, T. R. (1995). Towards principles for the design of ontologies used for knowledge sharing. *International Journal Human-Computer Studies*, 43:907–928.

Hagen, P. E. (2001). AUV/UUV mission planning and real time control with the HUGIN operator systems. In *Proceedings of the IEEE International Conference Oceans (Oceans'01)*, volume 1, pages 468–473, Honolulu, HI, USA.

Hagen, P. E. (2008). Applications of AUVs with SAS. In *Proceedings of the 10th Unmanned Underwater Vehicle Showcase (UUVS'08)*, Southampton, UK.

Hamilton, K. (2002). *An Integrated Diagnostic Architecture for Autonomous Robots*. PhD thesis, Heriot-Watt University.

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal 29*, 2:147–160.

Hanks, S. and Weld, D. S. (1994). A domain-independent algorithm for plan adaptation. *J. Artif. Int. Res.*, 2(1):319–360.

Horty, J. F. and Pollack, M. E. (2001). Evaluating new options in the context of existing plans. *Artificial Intelligence*, 127:200–1.

Howey, R., Long, D., and Fox, M. (2004). Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. pages 294–301.

Huang, H.-M., Pavek, K., Albus, J., and Messina, E. (2005). Autonomy levels for unmanned systems (alfus) framework: An update. In *Proceedings of the 2005 SPIE Defense and Security Symposium*, Orlando, Florida.

Hunt, J. W. and McIlroy, M. D. (1976). An algorithm for differential file comparison. *Computing Science Technical Report, Bell Laboratories 41*.

Jakuba, M. (2007). *Stochastic Mapping for Chemical Plume Source Localization with Application to Autonomous Hydrothermal Vent Discovery*. Doctor of philosophy in mechanical engineering, MIT/WHOI Joint Program in Applied Ocean Physics and Engineering.

Jamieson, J. and Tena, I. (2009). Autonomous vehicle meets new challenges. In *Deep Offshore Technology Conference (DOT'09)*, New Orleans, LA, USA.

Johnson, N., Patrón, P., and Lane, D. M. (2007). The importance of trust between operator and AUV: Crossing the human/computer language barrier. In *Proceedings of the IEEE International Conference Oceans Europe (Oceans Europe'07)*, Aberdeen, UK.

Jónsson, A. K., Morris, P. H., Muscettola, N., Rajan, K., and Smith, B. D. (2000). Planning in interplanetary space: Theory and practice. In *Artificial Intelligence Planning Systems*, pages 177–186.

JRP Master Plan (2005). Joint robotics program master plan FY2005. Technical report, US Department of Defense.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

Kambhampati, S. and Hendler, J. A. (1992). A validation-structure-based theory of plan modification and reuse. volume 55, pages 193–258, Essex, UK. Elsevier Science Publishers Ltd.

Kendall, M. (1938). A new measure of rank correlation. *Biometrika*, 30(1–2):81–89.

Kilfoyle, D. B. and Baggeroer, A. B. (2000). The state of the art in underwater acoustic telemetry. *IEEE Journal of Oceanic Engineering*, 25:4–27.

Knight, R., Fisher, F., Estlin, T., Engelhardt, B., and Chien, S. (2001). Balancing deliberation and reaction, planning and execution for space robotic applications. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'01)*, Maui, HI.

Koenig, S., Likhachev, M., and Furcy, D. (2004). Lifelong planning A*. *Artificial Intelligence*, 155(1-2):93–146.

Kokar, M. M., Matheus, C. J., and Baclawski, K. (2009). Ontology-based situation awareness. *Information Fusion*, 10(1):83–98.

Krogt, R. V. D. and Weerdt, M. D. (2005). Plan repair as an extension of planning. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, pages 161–170.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge University Press.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710.

Lynn, L. (2001). Forecasting critical military and commercial technologies: Potential long-term challenges for export controls. Technical report, The Henry L. Stimson Center.

Marine Metadata Interoperability (2008). Marine metadata interoperability. http://www.marinemetadata.org/.

Matheus, C., Kokar, M., and Baclawski, K. (2003). A core ontology for situation awareness. *Proceedings of the Sixth International Conference of Information Fusion*, 1:545 – 552.

McGann, C., Py, F., Rajan, K., and Henthorn, R. (2008a). Adaptive control for autonomous underwater vehicles. In *Association for the Advancement of Artificial Intelligence (AAAI'08)*, Chicago, IL, USA.

McGann, C., Py, F., Rajan, K., Henthorn, R., and McEwen, R. (2008b). A deliberative architecture for AUV control. In *International Conference on Robotic and Autonomation (ICRA'08)*, Pasadena, CA, USA.

McGann, C., Py, F., Rajan, K., and Olaya, A. G. (2009). Integrated planning and execution for robotic exploration. In *International Workshop on Hybrid Control of Autonomous Systems*, Pasadena, CA, USA.

McGann, C., Py, F., Rajan, K., Thomas, H., Henthorn, R., and McEwen, R. (2007). T-REX: A model-based architecture for AUV control. In *Workshop in Planning and Plan Execution for Real-World Systems: Principles and Practices for Planning in Execution, International Conference of Autonomous Planning and Scheduling (ICAPS'07)*.

Mechanical Turk (2010). Mechanical turk is a marketplace for work. http://www.mturk.com/mturk/welcome/.

Microsoft Robotic Developer Studio (2009). User guide. Technical report, Microsoft.

Miguelanez, E., Lewis, R., K.Brown, Roberts, C., and Lane, D. (2008). Fault diagnosis of a train door system based on the semantic knowledge representation. In *The 4th IET International Conference on Railway Condition Monitoring RCM 2008*, pages 1 –6, Derby Conference Centre, Derby, UK.

Miguelanez, E., Patrón, P., Brown, K., Petillot, Y. R., and Lane, D. M. (2010). Semantic knowledge-based framework to improve the situation awareness of autonomous underwater vehicles. *IEEE Transactions on Knowledge and Data Engineering (In Press)*, PP(99).

Miller, C. (2005). Trust in adaptive automation: The role of etiquette in tuning trust via analogic and affective method. In *Proceedings of the 1st International Conference on Augmented Cognition*, pages 22–27.

Morley, D. N., Myers, K. L., and Yorke-Smith, N. (2006). Continuous refinement of agent resource estimates. page 858865.

Murton, B. J. (2000). A global review of non-living resources on the extended continental shelf. *Revista Brasileira de Geofisica*, 18(3):281–306.

Myers, K. L. (1998). Towards a framework for continuous planning and execution. In *AAAI Fall Symposium on Distributed Continual Planning*.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys (CSUR) archive*, 33(1):31–88.

Nebel, B. and Koehler, J. (1995). Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence*, 76:427–454.

Newman, P. (2002). MOOS – a mission oriented operating suite. Technical report, Department of Ocean Engineering, Massachusetts Institute of Technology.

Newman, P. (2009a). Introduction to programming with MOOS. Technical report, Oxford Robotics Research Group.

Newman, P. (2009b). Under the hood of the MOOS communications API. Technical report, Oxford Robotics Research Group.

Newman, P., Westwood, R., and Westwood, J. (2007). Market prospects for AUVs. *Hydro International, ISSN: 1385-4569*, 11(10).

Niles, I. and Pease, A. (2001). Towards a standard upper ontology. *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*.

Ocean Observatories Initiative (2000). llluminating the hidden planet: The future of seafloor observatory science. Technical report, Committee on Seafloor Observatories: Challenges and Opportunities, Ocean Studies Board, National Research Council, National Academy Press, Washington, D.C. 135 pp. ISBN-13: 978-0-309-09032-2.

OceanSHELL (2005). OceanSHELL: An embedded library for distributed applications and communications. Technical report, Ocean Systems Laboratory, Heriot-Watt University.

Oliveira, P., Pascoal, A., Silva, V., and Silvestre, C. (1998). Mission control of the MARIUS AUV: System design, implementation, and sea trials. *International Journal of Systems Science*, 29(10):1065–1085.

O'Reilly, T. (2009). MBARI PUCK specification v1.3. Technical report, Monterey Bay Aquarium Research Institute.

Pailhas, Y., Capus, C., Brown, K., and Moore, P. (2010). Analysis and classification of broadband echoes using bio-inspired dolphin pulses. *Accepted for publication at the Journal of Acoustical Society of America*, 127(5).

Palomeras, N., Carreras, M., Ridao, P., and Hernandez, E. (2006). Mission control system for dam inspection with an AUV. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06)*, pages 2551–2556.

Palomeras, N., Ridao, P., Carreras, M., and Silvestre, C. (2009). Using petri nets to specify and execute missions for autonomous underwater vehicles. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*.

Pang, S., Farrell, J., Arrieta, R., and Li, W. (2003). AUV reactive planning: deepest point. In *Proceedings of the IEEE International Conference Oceans (Oceans'03)*, volume 4, pages 2222–2226.

Patrón, P. (2009). Embedded knowledge and autonomous planning. *Sea Technology Magazine, ISSN. 0093-3651*, 50(4):101.

Patrón, P. and Birch, A. (2009). Plan proximity: an enhanced metric for plan stability. In *Workshop on Verification and Validation of Planning and Scheduling Systems, 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 74–75, Thessaloniki, Greece.

Patrón, P., Evans, J., Brydon, J., and Jamieson, J. (2006a). AUTOTRACKER: Autonomous pipeline inspection: Sea trials 2005. In *Proceedings of the World Maritime Technology Conference - Advances in Technology for Underwater Vehicles (WMTC'06)*, London, UK.

Patrón, P., Evans, J., and Lane, D. M. (2006b). Fault tolerant decision making for unmanned vehicles. In *Proceedings of the Conference of Systems Engineering for Autonomous Systems from the Defence Technology Centre (SEAS-DTC'06)*, Edinburgh, UK.

Patrón, P., Evans, J., and Lane, D. M. (2007a). Mission plan recovery for increasing vehicle autonomy. In *Proceedings of the Conference of Systems Engineering for Autonomous Systems from the Defence Technology Centre (SEAS-DTC'07)*, Edinburgh, UK.

Patrón, P. and Lane, D. M. (2008). Adaptive mission planning: The embedded OODA loop. In *Proceedings of the Conference of Systems Engineering for Autonomous Systems from the Defence Technology Centre (SEAS-DTC'08)*, Edinburgh, UK.

Patrón, P., Lane, D. M., and Petillot, Y. R. (2009a). Continuous mission plan adaptation for autonomous vehicles: balancing effort and reward. In *4th Workshop on Planning and Plan Execution for Real-World Systems, 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 50–57, Thessaloniki, Greece.

Patrón, P., Lane, D. M., and Petillot, Y. R. (2009b). Interoperability of agent capabilities for autonomous knowledge acquisition and decision making in unmanned platforms. In *Proceedings of the IEEE International Conference Oceans Europe (Oceans Europe'09)*, Bremen, Germany.

Patrón, P., Lane, D. M., and Petillot, Y. R. (2009c). Situation-aware mission planning using distributed service oriented agents in autonomous underwater vehicles. In *International Symposium on Unmanned Untethered Submersible Technology*, Durham, NH, USA.

Patrón, P., Miguelanez, E., Cartwright, J., and Petillot, Y. R. (2008a). Semantic knowledge-based representation for improving situation awareness in service oriented agents of autonomous underwater vehicles. In *Proceedings of the IEEE International Conference Oceans (Oceans'08)*, Quebec, Canada.

Patrón, P., Miguelanez, E., Petillot, Y. R., and Lane, D. M. (2008b). Fault tolerant adaptive mission planning with semantic knowledge representation for autonomous underwater vehicles. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08)*, pages 2593–2598, Nice, France.

Patrón, P., Miguelanez, E., Petillot, Y. R., Lane, D. M., and Salvi, J. (2008c). Adaptive mission plan diagnosis and repair for fault recovery in autonomous underwater vehicles. In *Proceedings of the IEEE International Conference Oceans (Oceans'08)*, Quebec, Canada.

Patrón, P. and Petillot, Y. R. (2008). The underwater environment: A challenge for planning. In *Proceedings of the Conference of the UK Planning Special Interest Group (PlanSIG'08)*, Edinburgh, UK.

Patrón, P., Smith, B., Pailhas, Y., Capus, C., and Evans, J. (2005). Strategies and sensors technologies for UUV collision, obstacle avoidance and escape. In *7th Unmanned Underwater Vehicle Showcase (UUVS'05)*, Southampton, UK.

Patrón, P., Smith, B., Pailhas, Y., Capus, C., and Evans, J. (2007b). Strategies and sensor technologies for UUV collision, obstacle avoidance and escape. *Journal of the Undersea Defence Technology Forum*, 1:31–36.

Patrón, P. and Tena-Ruiz, I. (2006). A smooth simultaneous localisation and mapping solution to improve situational awareness, mission planning and re-planning for AUVs. In *Proceedings of the World Maritime Technology Conference - Advances in Technology for Underwater Vehicles (WMTC'06)*, London, UK.

Pearson, K. (1920). Notes on the history of correlation. *Biometrika*, 13:25–45.

Pell, B., Gat, E., Keesing, R., Muscettola, N., and Smith, B. (1997). Robust periodic planning and execution for autonomous spacecraft. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence (IJCAI'97)*, pages 1234–1239.

Penberthy, J. and Weld, D. S. (1992). UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the National Conference on Knowledge Representation and Reasoning (KR)*, pages 103–114.

Petillot, Y. R., Sotzing, C., Patrón, P., Lane, D. M., and Cartwright, J. (2009). Multiple system collaborative planning and sensing for autonomous platforms with shared and distributed situational awareness. In *Proceedings of the AUVSIs Unmanned Systems Europe*, La Spezia, Italy.

Pêtrès, C., Pailhas, Y., Patrón, P., Evans, J., Petillot, Y., and Lane, D. (2009). *Underwater Vehicles*, chapter 21:Trajectory Planning for Autonomous Underwater Vehicles, pages 399–416. I-Tech, ISBN 978-953-7619-49-7.

Pêtrès, C., Pailhas, Y., Patrón, P., Petillot, Y. R., Evans, J., and Lane, D. M. (2007). Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics*, 23(2):331–341.

Pêtrès, C. and Patrón, P. (2005). Path planning for unmanned underwater vehicles. In *Workshop on Planning and Learning in A Priori Unknown or Dynamic Domains, 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, Edinburgh, UK.

Platts, J. (2006). Autonomous systems design – a human centric paradox. In *MIT Humans and Technology Symposium*.

Portele, C. (2007). OpenGIS geography markup language (GML) encoding standard v.3.2.1. Technical report, Open Geospatial Consortium Inc.

Rabideau, G., Knight, R., Chien, S., Fukunaga, A., and Govindjee, A. (1999). Iterative repair planning for spacecraft operations in the ASPEN systems. In *Proceedings of the Fifth International Symposium on Artificial Intelligence Robotics and Automation in Space*, pages 99–106.

Rajan, K., McGann, C., Py, F., and Thomas, H. (2007). Robust mission planning using deliberative autonomy for autonomous underwater vehicles. In *Workshop in Robotics in challenging and hazardous environments, International Conference on Robotics and Automation (ICRA'07)*.

Rajan, K., Py, F., McGann, C., Ryan, J., O'Reilly, T., Maughan, T., and Roman, B. (2009). Onboard adaptive control of AUVs using automated planning and execution. In *International Symposium on Unmanned Untethered Submersible Technology (UUST'09)*, Durham, NH, USA.

Reed, S., Cormack, A., Hamilton, K., Ruiz, I. T., and Lane, D. (2006a). Automatic ship hull inspection using unmanned underwater vehicles. In *Proceedings from the 7th International Symposium on Technology and the Mine Problem*.

Reed, S., Petillot, Y., and Bell, J. (2003). An automatic approach to the detection and extraction of mine features in sidescan sonar. *Oceanic Engineering, IEEE Journal of*, 28(1):90 – 105.

Reed, S., Ruiz, I., Capus, C., and Petillot, Y. (2006b). The fusion of large scale classified side-scan sonar image mosaics. *IEEE Transactions on Image Processing*, 15(7):2049–2060.

REMUS AUV RECON v1.12 (2008). REMUS remote control protocol v1.12. Technical report, Hydroid Inc.

Rhodes, M. and Hyland, B. (2009). Underwater communications. Technical report, Patent application number: 20090245025.

Richards, D. and Howitt, S. (2006). Cognitive engineering and HMI design of a UAV ground control station. In *Third Annual Workshop on Human Factors of Unmanned Aerial Vehicles*.

Ridao, P., Batlle, J., Amat, J., and Roberts, G. (1999). Recent trends in control architectures for autonomous underwater vehicles. *International Journal of Systems Science*, 30(9):1033–1056.

Rosenblatt, J. K., Williams, S. B., and Durrant-Whyte, H. (2002). Behavior-based control for autonomous underwater exploration. *International Journal of Information Sciences,*, 145(1–2):69–87.

Sacerdoti, E. (1975). The nonlinear nature of plans. In *International Joint Conference on Artificial Intelligence (IJCAI'75)*, pages 206–214.

SAE-AS-4 AIR5665 (2008). Society of automotive engineers AS-4 AIR5665 JAUS architecture framework for unmanned systems. Technical report, SAE International Group.

SAE-AS-4 AS5684 (2008). Society of automotive engineers AS-4 AS5684 JAUS service interface definition language. Technical report, SAE International Group.

SAE-AS4 AIR5664 (2006). Society of automotive engineers AS-4 AIR5664 JAUS history and domain model. Technical report, SAE International Group.

SAUCE (2009). Mission rules for the student autonomous underwater challenge 2009 - europe v.1. Technical report, Defence Science and Technology Laboratory - Ministry of Defence, UK.

Saul, D. and Tena, I. (2007). BP's AUV development program, long term goals - short term wins. In *Proceedings of the IEEE International Conference Oceans (Oceans'07)*, pages 1–5.

Schulz, B., Hughes, R., Matson, E., Moody, R., and Hobson, B. (2005). The development of a free-swimming UUV for mine neutralization. In *Proceedings of the IEEE International Conference Oceans (Oceans'05)*, volume 2, pages 1443–1447.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53.

Somby, M. (2007). A review of robotics software platforms. http://www.windowsfordevices.com/c/a/Windows-For-Devices-Articles/A-review-of-robotics-software-platforms/.

Spearman, C. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, 15:72–101.

Stentz, A. (1994). The D* algorithm for real-time planning of optimal traverses. Technical Report CMU-RI-TR-94-37, Robotics Institute, Carnegie Mellon University.

Stojanovic, M. (2003). Acoustic (underwater) communications. In *Encyclopedia of Telecommunications, John G. Proakis*. Ed. John Wiley & Sons.

Tanenbaum, A. S. (1979). *Structured Computer Organization*. Prentice-Hall, ISBN: 0-13-148521-0.

Thorhallsson, T. and Hardason, H. (2003). GAVIA – a modular compact AUV for deep and shallow waters. In *Proceedings of the Unmanned Underwater Vehicle Showcase (UUVS'03)*, pages 45–58, New Malden, UK. Spearhead Exhibitions.

Thornton, J. (2005). Survivability - its importance in the maritime environment. *Journal of Defence Science*, 10(2):57–60.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT Press, ISBN: 0-262-20162-3.

Todo, Y., Kitazato, H., Hashimoto, J., and Gooday, A. J. (2005). Simple foraminifera flourish at the ocean's deepest point. *Science*, 307(5710):689.

Turner, R. (2005). Intelligent mission planning and control of autonomous underwater vehicles. In *Workshop on Planning under uncertainty for autonomous systems, 15th International Conference on Automated Planning and Scheduling (ICAPS'05)*.

Turner, R. M. (1995). Context-sensitive, adaptive reasoning for intelligent auv control: Orca project update. In *In Proceedings of the Ninth International Symposium on Unmanned, Untethered Submersible Technology (UUST'95)*, pages 426–435, Durham, NH, USA.

US DoD RoadMap (2009). Unmanned systems integrated roadmap FY2009–2034. Technical report, US Departement of Defense.

UUV Master Plan (2004). The navy unmanned undersea vehicle (UUV) master plan. Technical report, US Department of the Navy.

van der Krogt, R. (2005). *Plan repair in single-agent and multi-agent systems*. PhD thesis, Netherlands TRAIL Research School.

van Heijst, G., Schreiber, A., and Wielinga, B. (1996). Using explicit ontologies in kbs development. *International Journal of Human-Computer Studies*, 46(2-3).

Veloso, M. M. and Carbonell, J. G. (1993). Derivational analogy in prodigy: Automating case acquisition, storage, and utilization. volume 10, pages 249–278, Hingham, MA, USA. Kluwer Academic Publishers.

von Alt, C. (2010). Remus technology 2020 and beyond. In *NATO Naval Armaments Group – Maritime Capability Group 3 on Mines, Mine Countermeasures and Harbour Protection – Industry Day*, Porton Down, UK.

Wilkins, D. E. and desJardins, M. (2001). A call for knowledge-based planning. *AI Magazine*, 22(1):99–115.

Wilkins, D. E., Smith, S., Kramer, L., Lee, T. J., and Rauenbusch, T. W. (2005). Execution monitoring and replanning with incremental and collaborative scheduling. In *Workshop on Multiagent Planning and Scheduling, The 15th International Conference on Automated Planning & Scheduling (ICAPS'01)*.

Willow Garage ROS (2009). Introduction to ROS. Technical report, Willow Garage.

Yamamoto, I., Aoki, T., Tsukioka, S., Yoshida, H., Hyakudome, T., Sawa, T., Ishibashi, S., Inada, T., Yokoyama, K., Maeda, T., lshiguro, S., Hirayama, H., Hirokawa, K., Hashimoto, A., Hisatome, N., and Tani, T. (2004). Fuel cell system of AUV Urashima. In *Proceedings of the IEEE International Conference Oceans (Oceans'04)*, volume 3, pages 1732–1737.

Yu, X. (2000). Wireline quality underwater wireless communication using high speed acoustic modems. In *Proceedings of the IEEE International Conference Oceans (Oceans'00)*, volume 1, pages 417–422.

Yuh, J. (2000). Design and control of autonomous underwater robots: A survey. *Journal of Autonomous Robots*, 8(1):7–24.